

MENU

SEARCH

INDEX

DETAIL

1/1



JAPANESE PATENT OFFICE

PATENT ABSTRACTS OF JAPAN

(11)Publication number: 10091443

(43)Date of publication of application: 10.04.1998

(51)Int.Cl.

G06F 9/42
G06F 9/30
G06F 9/46

(21)Application number: 09135923

(71)Applicant:

SEIKO EPSON CORP

(22)Date of filing: 08.05.1997

(72)Inventor:

KUBOTA SATORU

KUDO MAKOTO

MIYAYAMA YOSHIYUKI

SATO HISAO

(30)Priority

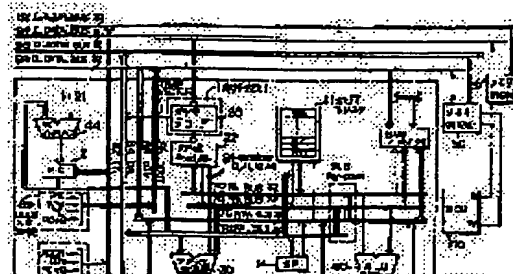
Priority number: 08127541 Priority date: 22.05.1996 Priority country: JP

(54) INFORMATION PROCESSING CIRCUIT, MICROCOMPUTER AND ELECTRONIC EQUIPMENT

(57)Abstract:

PROBLEM TO BE SOLVED: To efficiently describe a processing for dealing with a stack pointer with short instruction length, to efficiently describe the processings of execution, register saving and register restoration and to improve the processing speed of an interruption processing and sub-routine call return.

SOLUTION: CPU 10 decodes a stack pointer-only instruction group which contains a register SP14 only for the stack pointer and sets SP14 to be a mute operand by an instruction decoder 20. The stack pointer-only instruction group is executed in terms of



THIS PAGE BLANK (USPTO)

hardware by using a generalpurpose register 11, PC12, SP14, an address adder 30, ALU 40 a PC incrementor 44, internal buses 72, 74, 76 and 78, internal signal lines 82, 84, 86 and 88 and external buses 92, 94, 96 and 98. The stack pointer-only instruction group contains an SP relative load instruction, a stack pointer shift instruction, a call instruction, a return instruction, a continuous push instruction and a continuous pop instruction.

LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

Copyright (C); 1998 Japanese Patent Office

[MENU](#)

[SEARCH](#)

[INDEX](#)

[DETAIL](#)

THIS PAGE BLANK (USPTO)

(51) Int.Cl.⁶
 G 0 6 F 9/42 3 3 0
 9/30 3 5 0
 9/46 3 1 3

F I
 G 0 6 F 9/42 3 3 0 A
 9/30 3 5 0 B
 9/46 3 1 3 Z

審査請求 未請求 請求項の数17 F D (全 29 頁)

(21) 出願番号 特願平9-135923

(22) 出願日 平成9年(1997) 5月8日

(31) 優先権主張番号 特願平8-127541

(32) 優先日 平8(1996) 5月22日

(33) 優先権主張国 日本 (J P)

(71) 出願人 000002369

セイコーエプソン株式会社

東京都新宿区西新宿2丁目4番1号

(72) 発明者 久保田 哲

長野県諏訪市大和3丁目3番5号 セイコ

ーエプソン株式会社内

(72) 発明者 工藤 真

長野県諏訪市大和3丁目3番5号 セイコ

ーエプソン株式会社内

(72) 発明者 宮山 芳幸

長野県諏訪市大和3丁目3番5号 セイコ

ーエプソン株式会社内

(74) 代理人 弁理士 井上 一 (外2名)

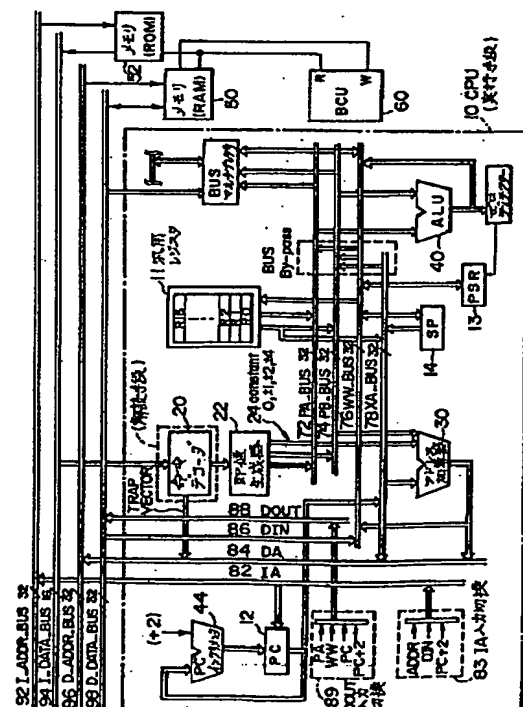
最終頁に続く

(54) 【発明の名称】 情報処理回路、マイクロコンピュータ及び電子機器

(57) 【要約】

【課題】 スタックポインタを取り扱う処理を、短い命令長で効率よく記述し、実行すること及びレジスタ退避やレジスタ復旧の処理を、効率よく記述し、割り込み処理及びサブルーチンコール・リターン処理の処理速度を向上させること。

【解決手段】 CPU10は、スタックポインタ専用のレジスタSP14を含みSP14を暗黙のオペランドとするスタックポインタ専用命令群を命令デコーダ20で解読する。そして、汎用レジスタ11、PC12、SP14、アドレス加算器30、ALU40、PCインクリメンタ44、内部バス72、74、76、78、内部信号線82、84、86、88、外部バス92、94、96、98等を用いて前記スタックポインタ専用命令群をハードウェア的に実行する。前記スタックポインタ専用命令群は、SP相対ロード命令やスタックポインタ移動命令やcall命令やret命令や連続プッシュ命令や連続ポップ命令を含む。



【特許請求の範囲】

【請求項1】 スタックポインタ専用用いるスタックポインタ専用レジスタと、
該スタックポインタ専用レジスタを暗黙のオペランドとするオブジェクトコードを有し、該スタックポインタ専用レジスタに基づき処理が記述されたスタックポインタ専用命令群のオブジェクトコードを解読して、該オブジェクトコードに基づき制御信号を出力する解読手段と、
前記スタックポインタ専用命令群を、前記制御信号及び前記スタックポインタ専用レジスタの内容に基づき実行する実行手段とを含むことを特徴とする情報処理回路。

【請求項2】 請求項1において、
前記スタックポインタ専用命令群が、転送レジスタ特定情報をオブジェクトコードに有するロード命令を含み、
前記解読手段が、
前記ロード命令を解読し、
前記実行手段が、
前記ロード命令を実行する際、メモリ上の所与の第一のエリアから所与の第一のレジスタへのデータの転送及び前記所与の第一のレジスタから前記所与の第一のエリアへのデータの転送の少なくとも一方を、前記スタックポインタ専用レジスタにより特定されるメモリアドレス及び前記転送レジスタ特定情報により特定されるレジスタアドレスに基づき行うことを特徴とする情報処理回路。

【請求項3】 請求項2において、
前記ロード命令が、前記メモリ上の前記第一のエリアのアドレスを特定するためのオフセットに関する情報であるオフセット情報をオブジェクトコードに含み、
前記実行手段が、
前記スタックポインタ専用レジスタの内容と前記オフセット情報により前記メモリアドレスを特定することを特徴とする情報処理回路。

【請求項4】 請求項3において、
前記オフセット情報が、即値で与えられた即値オフセット情報とメモリ上の所与のデータのサイズに関するデータサイズ情報とを含み、
前記実行手段が、
前記即値オフセット情報と前記データサイズ情報とに基づき、前記即値オフセット情報を左論理シフトしてオフセット値を生成し、前記スタックポインタ専用レジスタの内容と前記オフセット値を加算した値により前記メモリアドレスを特定することを特徴とする情報処理回路。

【請求項5】 請求項1～請求項4のいずれかにおいて、
前記スタックポインタ専用命令群が、オブジェクトコードに移動情報を有しスタックポインタを移動するためのスタックポインタ移動命令を含み、
前記解読手段が、
前記スタックポインタ移動命令を解読し、
前記実行手段が、

前記スタックポインタ移動命令を実行する際、前記移動情報に基づき、前記スタックポインタ専用レジスタの内容を変更することを特徴とする情報処理回路。

【請求項6】 請求項5において、
前記移動情報が、即値で与えられた即値移動情報を含み、
前記命令実行手段が、
前記即値移動情報と前記スタックポインタ専用レジスタの内容とを加算する処理及び前記スタックポインタ専用レジスタの内容から前記即値移動情報を減算する処理の少なくとも一方の処理を行うことを特徴とする情報処理回路。

【請求項7】 請求項1～請求項6のいずれかにおいて、
連続して順序づけられた複数のレジスタを含み、
前記スタックポインタ専用命令群が、複数レジスタ特定情報をオブジェクトコードに有する連続プッシュ命令及び連続ポップ命令の少なくとも一方を含み、
前記解読手段が、
前記連続プッシュ命令及び連続ポップ命令の少なくとも一方を解読し、
前記命令実行手段が、
前記連続プッシュ命令及び前記連続ポップ命令の少なくとも一方を実行する際、前記複数のレジスタからメモリに設けられたスタックへ連続して複数回プッシュする処理及び前記スタックから前記複数のレジスタに連続して複数回ポップする処理の少なくとも一方を、前記スタックポインタ専用レジスタの内容により特定されるメモリアドレスに及び前記複数レジスタ特定情報とに基づき行うことを特徴とする情報処理回路。

【請求項8】 請求項7において、
0からn-1までのレジスタ番号で特定されたn個の汎用レジスタを含み、
前記連続プッシュ命令及び連続ポップ命令の少なくとも一方のオブジェクトコードが、前記複数レジスタ特定情報として、前記レジスタ番号のいずれかが指定された最終レジスタ番号を含み、
前記実行手段が、

レジスタ0から前記最終レジスタ番号で特定されるレジスタまでの複数のレジスタからメモリに設けられたスタックへ連続して複数回プッシュする処理及び前記スタックから前記複数のレジスタに連続して複数回ポップする処理の少なくとも一方を、前記スタックポインタ専用レジスタの内容により特定されるメモリアドレスに基づき行うことを特徴とする情報処理回路。

【請求項9】 請求項7又は8のいずれかにおいて、
前記実行手段が、
前記複数のレジスタのいずれか所与のレジスタの内容を、前記スタックポインタ専用レジスタで特定されるメモリアドレスに基づきメモリに設けられたスタックに書

き込む書き込み手段と、

前記書き込み手段による前記スタックへの書き込み回数をカウントする書き込み回数カウント手段と、

前記カウント手段によってカウントされた前記書き込み回数と前記複数レジスタ特定情報の値を比較する比較手段とを含み、

前記書き込み手段が、

第一の入力と第二の入力を加算器で加算し書き込み先を特定するための書き込みメモリアドレスを生成する書き込みメモリアドレス生成手段と、

前記加算器の第一の入力が、連続専用命令の実行開始時にはスタックポインタ専用レジスタの内容となるように制御し、それ以降は書き込みアドレス生成手段によって生成された書き込みアドレスとなるよう制御する第一の入力制御手段と、

前記加算器の第二の入力に前記スタックから1ワードを書き込んだときのオフセット値を出力する第二の入力制御手段と、

前記複数レジスタ特定情報から前記書き込み回数を減じた値で特定されるレジスタの内容を、前記書き込みメモリアドレスに基づき前記スタックに書き込む手段とを含み、

前記比較手段の比較結果に基づき、複数のレジスタからの前記スタックへの書き込み及び書き込み終了を制御することを特徴とする情報処理回路。

【請求項10】 請求項7～9のいずれかにおいて、

前記命令実行手段が、

前記スタックポインタ専用レジスタで特定されるメモリアドレスに基づきメモリに設けられたスタックの内容を読み出し、前記複数のレジスタのいずれか所与のレジスタに格納する読み出し手段と、

前記読み出し手段による前記スタックからの読み出し回数をカウントする読み出し回数カウント手段と、

前記カウント手段によってカウントされた前記読み出し回数と前記複数レジスタ特定情報の値を比較する比較手段とを含み、

前記読み出し手段が、

第一の入力と第二の入力を加算器で加算し書き込み先を特定するための書き込みメモリアドレスを生成する書き込みメモリアドレス生成手段と、

前記加算器の第一の入力が、連続専用命令の実行開始時にはスタックポインタ専用レジスタの内容となるように制御し、それ以降は読み出しアドレス生成手段によって生成された読み出しアドレスとなるよう制御する第一の入力制御手段と、

前記加算器の第二の入力に前記スタックから1ワードを書き込んだときのオフセット値を出力する第二の入力制御手段と、

前記読み出しメモリアドレスに基づき前記スタックの内容を読み出し、前記書き込み回数に基づき特定されるレ

ジスタに格納する読み出し手段とを含み、

前記比較手段の比較結果に基づき、前記スタックの内容の読み出し及び読み出し終了を制御することを特徴とする情報処理回路。

【請求項11】 請求項1～請求項10のいずれかにおいて、

プログラムカウンタ専用のプログラムカウンタレジスタを含み、

10 前記スタックポインタ専用命令群が、サブルーチンへ分岐する命令及び前記サブルーチンからのリターン命令である分岐命令を含み、

前記解読手段が、

前記分岐命令を解読し、

前記命令実行手段が、

前記分岐命令を実行する際、メモリに設けられたスタックの所与の第二のエリアへの前記プログラムカウンタレジスタの内容の待避及び前記第二のエリアの内容のプログラムカウンタレジスタへの復帰の少なくとも一方を、前記スタックポインタ専用レジスタにより特定されるメモリアドレスに基づき行う手段と、

20 前記待避及び前記復帰に基づき前記スタックポインタ専用レジスタの内容を更新する手段とを含むことを特徴とする情報処理回路。

【請求項12】 連続して順序づけられた複数のレジスタと、いずれかの汎用レジスタに割り当てられたスタックポインタを含む情報処理回路であって、

複数レジスタ特定情報をオブジェクトコードに有する連続プッシュ命令及び連続ポップ命令の少なくとも一方の命令のオブジェクトコードを解読して、該オブジェクトコードに基づき制御信号を出力する手段と、

30 前記連続プッシュ命令及び前記連続ポップ命令の少なくとも一方を実行する際、前記複数のレジスタからメモリに設けられたスタックへ連続して複数回プッシュする処理及び前記スタックから前記複数のレジスタに連続して複数回ポップする処理の少なくとも一方を、前記制御信号及び前記スタックポインタ専用レジスタの内容により特定されるメモリアドレス及び前記複数レジスタ特定情報とに基づき行う手段を含むことを特徴とする情報処理回路。

40 【請求項13】 請求項1～請求項12のいずれかにおいて、RISC方式であることを特徴とする情報処理回路。

【請求項14】 請求項1～請求項13のいずれかにおいて、

固定長の命令を解読し、該命令に基づき実行処理を行うことを特徴とする情報処理回路。

【請求項15】 請求項1～請求項14のいずれかに記載の情報処理回路と記憶手段と外部との入出力を行う手段とを含むことを特徴とするマイクロコンピュータ。

【請求項16】 請求項15において、

前記スタックポインタに関連づけてオート変数の記憶領域が確保される構造を有する言語のプログラムが実行されることを特徴とするマイクロコンピュータ。

【請求項17】 請求項15又は16のいずれかに記載されたマイクロコンピュータを含むことを特徴とする電子機器。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、情報処理回路、前記情報処理回路を内蔵するマイクロコンピュータ、及び該マイクロコンピュータを用いて構成された電子機器に関する。

【0002】

【背景技術及び発明が解決しようとする課題】従来、32ビットのデータを処理できるRISC方式のマイクロコンピュータでは、32ビット幅の固定長命令が用いられていた。その理由は、固定長命令を用いると、可変長命令を用いる場合に比べ、命令のデコードに要する時間を短縮でき、また、マイクロコンピュータの回路規模を小さくすることが出来るからである。

【0003】ところが、32ビットのマイクロコンピュータにおいても、特に32ビットも必要としない命令も多い。従って全ての命令を32ビットで記述すると、命令に冗長な部分が生じる命令が多くなり、メモリーの使用効率が悪くなる。

【0004】そこで、本願の発明者は、制御回路を複雑にすることなくメモリーの使用効率を向上させるために、処理出来るデータのビット幅より短いビット幅の固定長命令を処理するマイクロコンピュータについての検討を行っていた。

【0005】しかし、例えば32ビット固定長の命令を単に16ビットの固定長にすると以下のような問題点が生じる。

【0006】即ちRISC方式のマイクロコンピュータでは処理及び命令セットの汎用性を重視するため、スタックポインタを取り扱う処理を行う場合、いずれかの汎用レジスタをスタックポインタとして用い、汎用レジスタを操作対象とする命令を用いて前記処理を行っていた。従って、このような処理を記述する際の前記命令の中には、スタックポインタとして使用している汎用レジスタの指定が必要となる。

【0007】例えば、スタックポインタに所与のオフセット値を加えたメモリアドレスで特定されるデータを所与のレジスタに転送する処理を汎用レジスタを操作対象とする命令で記述する場合、その命令のオブジェクトコードには、前記オフセット値、前記所与のレジスタを特定する情報、スタックポインタとして使用しているレジスタを特定する情報が必要となる。

【0008】このように、汎用レジスタを操作対象とする命令でスタックポインタを取り扱う処理を記述する場

合、オブジェクトコードで指定する情報が多くなるため、命令の内容を16ビットの固定長で記述することが困難となる。ここで命令長を例えば32ビットにすると、命令の中にも特に命令に32ビットも必要としない命令も多いため、命令に冗長な部分を生じる命令が多くなり、メモリーの使用効率の悪化を招く。

【0009】また、命令長が長くなればそれを格納するメモリーも余分に必要になるため、メモリーの効率的な使用という観点からは、固定長の命令に限らず、命令長を短くできることが好ましい。

【0010】また、例えばC言語のようにスタックポインタに関連づけてオート変数の記憶領域が確保される言語で記述されたプログラムを実行させる場合、スタックポインタを取り扱う命令が多くなるため、スタックポインタを取り扱う命令を効率的に記述し、実行させることが望ましい。

【0011】そこで、スタックポインタを取り扱う処理を行う場合、できるだけ短い命令長でその命令内容を記述し、実行することができるアーキテクチャが望まれていた。

【0012】また最近の特にRISC方式のCPUは、性能を高めるため内部に多くの汎用レジスタを持つようになった。多くのレジスタを内部に持つことによりメモリーにアクセスすることなくCPU内部で高速に多くの処理を出来るからである。このように内部レジスタを多く持つと、割り込み処理やサブルーチンコールの時のレジスタ退避と復旧の処理の際、退避すべきレジスタ数が多くなる。

【0013】以下、スタック系命令の中でもサブルーチンに入ったりでたりする際に多用するレジスタの待避、復帰命令を例にとり従来例を説明する。

【0014】通常マイクロコンピュータの命令セットはCPUのレジスタをメモリーに設けたスタックに待避したり復旧するための命令を持っている。そのための専用の命令を持つもの、あるいはレジスタ間接アドレッシング命令をもつものがある。

【0015】前記待避したり復旧したりするための命令に関する技術としては、インテル社の80386に関して、「80386プログラミング」(John H. Crawford Patric P. Gelsigner 著 岩谷宏 訳)に以下のような記述がある。

【0016】即ちレジスタをスタックに書き込む命令として push、pusha、pushadがあり、スタックからレジスタにデータを戻すための命令として pop、popa、popadがある。

【0017】push命令でレジスタをスタックに書き込むときは push EAX のようにオペランドとしてレジスタを指定する。これは32ビットレジスタEAXの場合である。レジスタEAX、ECX、EDX、EBXを全てスタックに書く場合は、

10

20

30

40

50


```
push EAX
push ECX
push EDX
push EBX
```

のようにpush命令を繰り返す。

【0018】このようにpush、pop命令でレジスタを1本ずつ操作しているとオブジェクトコードのサイズは大きくなり、プログラムの実行ステップも多くなり、そのためプログラムの実行時間や処理動作はより遅くなる。

【0019】そこで80386の有する8本の汎用レジスタ全てをスタックに書き込むにはpusha またはpushad 命令を使う。pushaは8本のレジスタの其々の下位16ビットレジスタを対象とし、pushadは32ビットを対象とする。pusha、pushadによりpush命令を8回繰り返すことを省略できる。

【0020】pop、popa、popad命令についても同様である。

【0021】push命令を繰り返すことの不利な点はプログラムコードが長くなること、1命令毎フェッチし実行するため実行が遅いことである。この点は、前記pusha、pushadにより、前記8本のレジスタを全てスタックに書き込む場合には大きく改善される。しかし、4本の場合とか6本の場合等のように8未満のレジスタを書き込む場合には、この点は改善されない。

【0022】即ち、80386のようにpusha、pushad、popa、popad命令で全てのレジスタを操作すると、レジスタの全てを退避・復旧する必要の無いときも、このサイクルの長い遅い命令を使わなければならない。このような場合、命令は一命令で済むがこの命令の実行サイクルは長くなるという問題が生じる。

【0023】また、call命令やリターン命令等でサブルーチンへ分岐したり、呼ばれたルーチンへ戻る際には、戻り先アドレスとして必要となるプログラムカウンタの待避や復旧の処理が必要となる。従来のRISC方式のCPUにおいては、これらの処理をソフトウェア的に実現していた。即ち、該処理を記述したアセンブラ命令(オブジェクトコード)を実行することにより、前記プログラムカウンタの待避や復旧の処理を行っていた。このため、call命令やリターン命令はオブジェクトコードの増加を招き、また1命令毎フェッチし実行するために実行速度の鈍化を招いていた。

【0024】本発明の目的は、スタックポインタを取り扱う処理を、短い命令長で効率よく記述し、実行することができるアーキテクチャを有する情報処理回路、マイクロコンピュータ、電子機器を提供することである。

【0025】また 本発明の他の目的は、レジスタ退避やレジスタ復旧の処理を、効率よく記述し、割り込み処

理及びサブルーチンコール・リターンの処理速度が速い情報処理回路、マイクロコンピュータ、電子機器を提供することである。

【0026】

【課題を解決するための手段】本発明は、スタックポインタ専用用いるスタックポインタ専用レジスタと、該スタックポインタ専用レジスタを暗黙のオペランドとするオブジェクトコードを有し、該スタックポインタ専用レジスタに基づく処理が記述されたスタックポインタ専用命令群のオブジェクトコードを解釈して、該オブジェクトコードに基づき制御信号を出力する解釈手段と、前記スタックポインタ専用命令群を、前記制御信号及び前記スタックポインタ専用レジスタの内容に基づき実行する実行手段とを含むことを特徴とする。

【0027】ここにおいてオブジェクトコードとは、一般に翻訳プログラムによって機械語に翻訳した結果得られるプログラムコードのことをいうが、本発明においては、翻訳プログラムによるかいかに関係なく、機械語で記述されているプログラムコードを含む広い概念で使用する。

【0028】本発明の情報処理回路は、スタックポインタ専用のスタックポインタ専用レジスタを有し、該スタックポインタ専用レジスタを操作対象とするスタックポインタ専用命令群の解釈、実行を行うよう構成されている。

【0029】前記スタックポインタ専用命令群は、スタックポインタ専用レジスタを取り扱うための専用のオペコードを有しているため、オブジェクトコードのオペランドにスタックポインタを特定するための情報を必要としない。言い換えれば、前記スタックポインタ専用命令群はスタックポインタ専用レジスタを暗黙のオペランドとしている。このため、汎用レジスタの1つをスタックポインタに割当て、汎用レジスタを操作対象とする命令を用いてスタックポインタの操作を行う場合に比べて、短い命令長でスタックポインタを取り扱う命令を記述することができる。

【0030】従って本発明によれば、スタックポインタを取り扱う処理を短い命令長で記述し実行することができる情報処理回路を提供することができる。また、命令を記憶するメモリの使用効率のよい情報処理回路を提供することができる。

【0031】本発明は、前記スタックポインタ専用命令群が、転送レジスタ特定情報をオブジェクトコードに有するロード命令を含み、前記解釈手段が、前記ロード命令を解釈し、前記実行手段が、前記ロード命令を実行する際、メモリ上の所与の第一のエリアから所与の第一のレジスタへのデータの転送及び前記所与の第一のレジスタから前記所与の第一のエリアへのデータの転送の少なくとも一方を、前記スタックポインタ専用レジスタにより特定されるメモリアドレス及び前記転送レジスタ特定

10

20

30

40

50

情報により特定されるレジスタアドレスに基づき行うことを特徴とする。

【0032】ここにおいて前記スタックポインタ専用命令群に含まれる前記ロード命令とは、メモリとレジスタ間でデータの転送を行う命令であり、メモリからレジスタへの転送及びレジスタからメモリへの転送の少なくとも一方を含む概念である。なお、データの内容は問わずアドレスデータも含む概念である。メモリアドレスとは、転送の際のメモリ上のエリアを特定するためのアドレスをいう。

【0033】前記ロード命令は、スタックポインタ専用レジスタを操作対象とするための専用のオペコードを有しているため、オブジェクトコードのオペランドにスタックポインタを特定するための情報を必要としない。このため、スタックポインタに関連づけたメモリアドレスを有するメモリ上のエリアとレジスタ間でデータの転送処理を行わせる場合、短い命令長で記述することができる。

【0034】本発明は、前記ロード命令が、前記メモリ上の前記第一のエリアのアドレスを特定するためのオフセットに関する情報であるオフセット情報をオブジェクトコードに含み、前記実行手段が、前記スタックポインタ専用レジスタの内容と前記オフセット情報により前記メモリアドレスを特定することを特徴とする。

【0035】ここにおいてオフセット情報とは、オフセット値を直接即値で指定したものでもよいし、オフセット値が格納されたレジスタ等のアドレスを指定する場合のように間接的に指定する場合でもよい。オフセット情報を含んだ前記ロード命令が実行される場合、転送の際に必要な前記メモリアドレスは、スタックポインタ専用レジスタの内容及びオフセット情報に基づき特定される。

【0036】従ってスタックポインタ及び前記オフセット情報に基づき特定されるメモリアドレスを有するメモリ上のエリアとレジスタ間でデータの転送処理を行わせる場合、短い命令長で記述することができる。

【0037】また本発明によれば、例えばスタックポインタが常にワード境界をさすような構造の情報処理回路においても、適当なオフセット情報を指定することにより、スタック上の任意のエリアを指定することが可能となる。このため、データのサイズに応じて効率良くスタックに格納することができ、スタックの使用効率の向上を図ることができる。

【0038】本発明は、前記オフセット情報が、即値で与えられた即値オフセット情報とメモリ上の所与のデータのサイズに関するデータサイズ情報とを含み、前記実行手段が、前記即値オフセット情報と前記データサイズ情報とに基づき、前記即値オフセット情報を左論理シフトしてオフセット値を生成し、前記スタックポインタ専用レジスタの内容と前記オフセット値を加算した値によ

り前記メモリアドレスを特定することを特徴とする。

【0039】ここにおいて即値オフセット情報とは、オフセット値を直接即値で指定したものをいう。またデータサイズ情報とは、転送すべきメモリ上データのサイズをいう。通常データサイズは、8ビットのバイトデータや、16ビットのハーフワードデータ、32ビットのワードデータ等の 2^n （ n は3以上）で表される。メモリ上のアドレスはバイト単位に与えられており、ハーフワードデータはハーフワード境界におかれ、ワードデータはワード境界におかれる。従って、ハーフワードデータのメモリアドレスの下位の1ビットは0となり、ワードデータのメモリアドレスの下位の2ビットは00となる。スタックポインタのアドレスは、ワード境界を指しているため、ハーフワードデータのメモリアドレスを生成する場合のオフセット値の下位の1ビットは0となり、ワードデータのメモリアドレスを生成する場合のオフセット値の下位の2ビットは00となる。

【0040】また、左論理シフトとは、データのビット列を左にずらし、シフトによってデータの右側にできあきビット（シフトインビット）に0が入るシフトをいう。

【0041】本発明によれば、データサイズに基づき、前記オフセット即値情報の左論理シフトを行うため、データサイズによって一義的に決まる下位のビットを省略して、即値オフセット情報を記述することができる。従って、即値オフセット情報を効率的に指定することができる。バイト以外のデータサイズの場合、そのまま指定する場合に比べて、より大きいオフセット値に指定が可能となる。

【0042】また該命令を用いることによって、データのメモリへの書き込み、読み出し時にそのデータサイズに応じた適切な境界位置が選択されることになる。

【0043】本発明は、前記スタックポインタ専用命令群が、オブジェクトコードに移動情報を有しスタックポインタを移動するためのスタックポインタ移動命令を含み、前記解読手段が、前記スタックポインタ移動命令を解読し、前記実行手段が、前記スタックポインタ移動命令を実行する際、前記移動情報に基づき、前記スタックポインタ専用レジスタの内容を変更することを特徴とする。

【0044】前記スタックポインタ移動命令は、スタックポインタ専用レジスタを操作対象とするための専用のオペコードを有しているため、オブジェクトコードのオペランドにスタックポインタを特定するための情報を必要としない。このため、スタックを移動させたい場合、短い命令長で記述することができる。従って、スタックに格納されているデータの処理やスタックポインタに関連づけて記憶されているデータの処理を行う際の命令の記述量を削減することができる。

【0045】本発明によればスタックの移動が容易に行

えるため、特に異なるルーチンにおいてそれぞれ異なるスタックエリアを確保して処理を行う場合に有効である。即ちルーチンごとにスタックポインタを移動させる処理を行うことで、広範囲な領域にわたるアドレス指定が可能となる。

【0046】本発明は、前記移動情報が、即値で与えられた即値移動情報を含み、前記命令実行手段が、前記即値移動情報と前記スタックポインタ専用レジスタの内容とを加算する処理及び前記スタックポインタ専用レジスタの内容から前記即値移動情報を減算する処理の少なくとも一方の処理を行うことを特徴とする。

【0047】本発明によれば、スタックポインタの値を即値移動情報で指定された分だけ上方又は下方に移動させる処理を短い命令長で記述することができる。

【0048】本発明は、連続して順序づけられた複数のレジスタを含み、前記スタックポインタ専用命令群が、複数レジスタ特定情報をオブジェクトコードに有する連続プッシュ命令及び連続ポップ命令の少なくとも一方を含み、前記解読手段が、前記連続プッシュ命令及び連続ポップ命令の少なくとも一方を解読し、前記命令実行手段が、前記連続プッシュ命令及び前記連続ポップ命令の少なくとも一方を実行する際、前記複数のレジスタからメモリに設けられたスタックへ連続して複数回プッシュする処理及び前記スタックから前記複数のレジスタに連続して複数回ポップする処理の少なくとも一方を、前記スタックポインタ専用レジスタの内容により特定されるメモリアドレスに及び前記複数レジスタ特定情報とに基づき行うことを特徴とする。

【0049】プッシュとはメモリに設けられたスタックにデータを積み重ねて格納すること、ポップとは前記スタックからデータを取り出すことをいう。ここにおいてプッシュする処理及びポップする処理とは、前記格納及び取り出しの処理と、それに伴うスタックポインタの更新処理を含む。通常の情報処理回路は1のレジスタからスタックにデータやアドレスを格納するためのpush命令、スタックの内容をレジスタに取り出すpop命令を有している。該push命令やpop命令は、レジスタとスタックとのデータのやり取りや、該やり取りに伴うスタックポインタの更新を行う。

【0050】従って複数のレジスタとスタックでデータ等のやり取りを行う場合には、これらの命令を複数回実行することが必要となる。

【0051】しかし本発明によれば、前記連続プッシュ命令や前記連続ポップ命令を実行すると、push命令を連続して複数回実行する場合やpop命令を連続して複数回実行する場合と同じ効果が得られる。即ち複数のレジスタとスタックとの間のデータのやり取り及び該やり取りに伴うスタックポインタの更新を1命令で実行することが出来る。このため複数のレジスタとスタックとの間のデータの転送を行う場合、push命令あるいは

pop命令をくり返すことによりオブジェクトコードサイズが増大するのを防ぐことができる。またプログラム実行ステップが長くなることを回避し、無駄なサイクルを消費することなく、割り込み処理及びサブルーチンコール・リターン処理の向上を図ることができる。

【0052】本発明は、0からn-1までのレジスタ番号で特定されたn個の汎用レジスタを含み、前記連続プッシュ命令及び連続ポップ命令の少なくとも一方のオブジェクトコードが、前記複数レジスタ特定情報として、前記レジスタ番号のいずれかが指定された最終レジスタ番号を含み、前記実行手段が、レジスタ0から前記最終レジスタ番号で特定されるレジスタまでの複数のレジスタからメモリに設けられたスタックへ連続して複数回プッシュする処理及び前記スタックから前記複数のレジスタに連続して複数回ポップする処理の少なくとも一方を、前記スタックポインタ専用レジスタの内容により特定されるメモリアドレスに基づき行うことを特徴とする。

【0053】通常汎用レジスタが複数ある場合、レジスタを特定するためのアドレスを有している。本発明では0からn-1までの連続したレジスタ番号で前記レジスタを特定している。

【0054】本発明によれば、前記最終レジスタ番号に任意のレジスタ番号を指定することにより、レジスタ0から前記最終レジスタ番号までの連続した複数のレジスタとメモリとの間でデータをプッシュする処理及びポップする処理の少なくとも一方が行われる。従って、レジスタ番号0のレジスタから順番にレジスタを使用するような構造を有するプログラムの実行において、レジスタの待避や復旧を効率的に行うことができる。

【0055】本発明は、前記実行手段が、前記複数のレジスタのいずれか所与のレジスタの内容を、前記スタックポインタ専用レジスタで特定されるメモリアドレスに基づきメモリに設けられたスタックに書き込む書き込み手段と、前記書き込み手段による前記スタックへの書き込み回数をカウントする書き込み回数カウント手段と、前記カウント手段によってカウントされた前記書き込み回数と前記複数レジスタ特定情報の値を比較する比較手段とを含み、前記書き込み手段が、第一の入力と第二の入力を加算器で加算し書き込み先を特定するための書き込みメモリアドレスを生成する書き込みメモリアドレス生成手段と、前記加算器の第一の入力が、連続専用命令の実行開始時にはスタックポインタ専用レジスタの内容となるように制御し、それ以降は書き込みアドレス生成手段によって生成された書き込みアドレスとなるよう制御する第一の入力制御手段と、前記加算器の第二の入力に前記スタックから1ワードを書き込んだときのオフセット値を出力する第二の入力制御手段と、前記複数レジスタ特定情報から前記書き込み回数を減じた値で特定されるレジスタの内容を、前記書き込みメモリアドレスに

基づき前記スタックに書き込む手段とを含み、前記比較手段の比較結果に基づき、複数のレジスタからの前記スタックへの書き込み及び書き込み終了を制御することを特徴とする。

【0056】また本発明は、前記命令実行手段が、前記スタックポインタ専用レジスタで特定されるメモリアドレスに基づきメモリに設けられたスタックの内容を読み出し、前記複数のレジスタのいずれか所与のレジスタに格納する読み出し手段と、前記読み出し手段による前記スタックからの読み出し回数をカウントする読み出し回数カウント手段と、前記カウント手段によってカウントされた前記読み出し回数と前記複数のレジスタ特定情報の値を比較する比較手段とを含み、前記読み出し手段が、第一の入力と第二の入力を加算器で加算し書き込み先を特定するための書き込みメモリアドレスを生成する書き込みメモリアドレス生成手段と、前記加算器の第一の入力が、連続専用命令の実行開始時にはスタックポインタ専用レジスタの内容となるように制御し、それ以降は読み出しアドレス生成手段によって生成された読み出しアドレスとなるよう制御する第一の入力制御手段と、前記加算器の第二の入力に前記スタックから1ワードを書き込んだときのオフセット値を出力する第二の入力制御手段と、前記読み出しメモリアドレスに基づき前記スタックの内容を読み出し、前記書き込み回数に基づき特定されるレジスタに格納する読み出し手段とを含み、前記比較手段の比較結果に基づき、前記スタックの内容の読み出し及び読み出し終了を制御することを特徴とする。

【0057】このようにすると連続した値で順序づけて特定される複数のレジスタのスタックへの待避又はスタックから連続した値で特定される複数のレジスタへの復旧をカウント手段と簡単なシーケンス制御のみで実現可能である。従って少ないゲート数の情報処理回路で実現出来るため、ワンチップのマイクロコンピュータ等に適したものである。

【0058】本発明は、プログラムカウンタ専用のプログラムカウンタレジスタを含み、前記スタックポインタ専用命令群が、サブルーチンへ分岐する命令及び前記サブルーチンからのリターン命令である分岐命令を含み、前記解読手段が、前記分岐命令を解読し、前記命令実行手段が、前記分岐命令を実行する際、メモリに設けられたスタックの所与の第二のエリアへの前記プログラムカウンタレジスタの内容の待避及び前記第二のエリアの内容のプログラムカウンタレジスタへの復帰の少なくとも一方を、前記スタックポインタ専用レジスタにより特定されるメモリアドレスに基づき行う手段と、前記待避及び前記復帰に基づき前記スタックポインタ専用レジスタの内容を更新する手段とを含むことを特徴とする。

【0059】ここにおいてサブルーチンとは、割り込み処理及び例外処理及びデバッグ処理ルーチン等も含む。従ってサブルーチンへ分岐する命令とは、サブルーチン

等をコールする命令、割り込み処理及び例外処理及びデバッグ処理ルーチン等へ分岐するためのソフトウェア割り込み命令、ソフトウェアデバッグ割り込み命令等も含む。またサブルーチンからのリターン命令には、割り込み処理及び例外処理及びデバッグ処理ルーチン等からのリターン命令を含む。

【0060】通常サブルーチンへ分岐する場合や前記サブルーチンから戻る場合プログラムカウンタの待避や復帰が必要となる。

【0061】本発明では、前記サブルーチンへ分岐する命令の実行及びサブルーチンからリターンする命令を実行する際に前記プログラムカウンタの待避及び復帰も同時に行う。即ち、本発明の情報処理回路は、サブルーチンへ分岐する命令及び前記サブルーチンからのリターン命令のいずれか一命令でプログラムカウンタの待避及び復帰が出来る回路構成を有している。このためサブルーチンへの分岐やサブルーチンからのリターンに伴って必要となる前記プログラムカウンタの待避及び復帰の命令が不要となり、命令数を削減することができる。また、無駄なサイクルを消費することなく、サブルーチンコール・リターン等の他のルーチンへ分岐時の処理速度の向上を図ることができる。

【0062】また、ソフトウェア割り込み命令が発生した場合には、例えばCPU等の情報処理回路の現在の状態を保持するプロセッサステータスレジスタの待避及び復帰も必要となる。従って、ソフトウェア割り込み命令等の場合は該命令の実行時にプロセッサステータスレジスタの待避及び復帰も同時に行うようにすることが好ましい。

【0063】本発明は、連続して順序づけられた複数のレジスタと、いずれかの汎用レジスタに割り当てられたスタックポインタを含む情報処理回路であって、複数レジスタ特定情報をオブジェクトコードに有する連続ブッシュ命令及び連続ポップ命令の少なくとも一方の命令のオブジェクトコードを解読して、該オブジェクトコードに基づき制御信号を出力する手段と、前記連続ブッシュ命令及び前記連続ポップ命令の少なくとも一方を実行する際、前記複数のレジスタからメモリに設けられたスタックへ連続して複数回ブッシュする処理及び前記スタックから前記複数のレジスタに連続して複数回ポップする処理の少なくとも一方を、前記制御信号及び前記スタックポインタ専用レジスタの内容により特定されるメモリアドレス及び前記複数のレジスタ特定情報とに基づき行う手段を含むことを特徴とする。

【0064】本発明は、汎用レジスタをスタックポインタとして使用する場合の前記連続ブッシュ命令及び前記連続ポップ命令に関する。

【0065】本発明によれば、前記連続ブッシュ命令や前記連続ポップ命令を実行すると、push命令を連続して複数回実行する場合やpop命令を連続して複数回

実行する場合と同じ効果が得られる。即ち複数のレジスタとスタックとの間のデータのやり取り及び該やり取りに伴うスタックポインタの更新を1命令で実行することが出来る。このため複数のレジスタとスタックとの間のデータの転送を行う場合、push命令あるいはpop命令をくり返すことによりオブジェクトコードサイズが増大するのを防ぐことができる。またプログラム実行ステップが長くなることを回避し、無駄なサイクルを消費することなく、割り込み処理及びサブルーチンコール・リターンの処理速度の向上を図ることができる。

【0066】本発明はの情報処理回路はRISC方式であることを特徴とする。

【0067】RISC方式の情報処理回路は、ハードウェアを小型化して高速化を図ることを目的として設計されている。このため汎用レジスタを多く有しており、命令セットを汎用性の高いものに絞ることで命令数の削減を図っている。

【0068】従ってRISC方式の情報処理回路では、スタックポインタを汎用レジスタに割り当て、スタックポインタを扱う場合は汎用レジスタを扱う命令セットを用いて処理を行っていた。しかしこの様な方法では命令長が大きくなりメモリの使用効率がよくない。

【0069】本発明によれば、RISC方式の情報処理回路において、命令長を削減することが出来、メモリの使用効率を上げることが出来る。

【0070】本発明の情報処理回路は、固定長の命令を解読し、該命令に基づき実行処理を行うことを特徴とする。

【0071】固定長命令を用いると可変長命令を用いる場合に比べ、命令のデコードに要する時間を短縮でき、情報処理回路の回路規模を小さくすることが出来る。固定長命令を採用する場合、命令に冗長な部分が出るのを防ぎメモリを効率よく使用するためには、各命令に必要なビット数はばらつきが少なく、出来るだけ短いほうが好ましい。

【0072】本発明によれば、一般に命令長が長くなりがちなスタックポインタを取り扱う命令の命令長を短くすることができる。従って固定長命令を採用した場合であっても命令に冗長な部分が出るのを防ぎ、メモリを効率よく使用することが出来る。

【0073】本発明のマイクロコンピュータは、前述した本発明の情報処理回路と記憶手段と外部との入出力を行う手段とを含むことを特徴とする。

【0074】本発明によれば、処理速度が速くメモリの使用効率のよい、マイクロコンピュータを提供することが出来る。

【0075】本発明のマイクロコンピュータは、前記スタックポインタに関連づけてオート変数の記憶領域が確保される構造を有する言語のプログラムが実行されることを特徴とする。

【0076】スタックポインタに関連づけてオート変数の記憶領域が確保される構造を有する言語として、例えばC言語がある。この様な言語のプログラムの処理を本発明のマイクロコンピュータが行う場合、処理速度及びメモリの使用効率を効果的に向上させることが出来る。

【0077】本発明の電子機器は、前述した本発明のマイクロコンピュータを含むことを特徴とする。

【0078】本発明によれば、処理速度が速くメモリの使用効率のよい情報処理回路を内蔵しているため、安価で高性能な電子機器を提供することが出来る。

【0079】

【発明の実施の形態】以下、本実施例を図面に基づき説明する。

【0080】(実施例1)

(1) 本実施例のCPUの構成

本実施例のCPUはパイプラインとロード・ストア型のアーキテクチャによって、ほとんど全ての命令を1サイクルで実行する。全ての命令は16ビットの固定長で記述されており、本実施例のCPUの処理する命令は極めて小さいオブジェクトコードサイズを実現している。

【0081】特に本実施例のCPUは、スタックポインタを取り扱う処理を効率よく記述し実行するためにスタックポインタ専用のレジスタを有し、該スタックポインタ専用レジスタを暗黙のオペランドとするオブジェクトコードを有するスタックポインタ専用命令群の命令セットを解読、実行出来るよう構成されている。

【0082】図1は、本実施例のCPUの回路構成の概略を説明するための図である。

【0083】本CPU10は、汎用レジスタ11、プログラムカウンタが格納されているPC12、プロセッサステータスレジスタ(PSR)13、スタックポインタ専用のレジスタであるSP14を含むレジスタセットと、命令デコーダ20、即値生成器22、アドレス加算器30、ALU40、PCインクリメンタ44及び各種内部バス72、74、76、78、各種内部信号線82、84、86、88等を含む。

【0084】前記命令デコーダ20は、入力したオブジェクトコードを解読し、該命令を実行するために必要な処理を行い、必要な制御信号を出力する。なお、該命令デコーダ20は前記スタックポインタ専用命令のオブジェクトコードを解読して該命令に基づき制御信号を出力する前記解読手段としても機能する。

【0085】即値生成器22は、オブジェクトコードに含まれた即値に基づき、実行時に使用する32ビットの即値データを生成したり、各命令の実行に必要な0、±1、±2、±4のconstantデータを生成したりする。PCインクリメンタ44は、命令の実行サイクルに基づきPC12に格納されたプログラムカウンタの更新を行う。アドレス加算器30は、各種レジスタに格納されて

いる情報や即値生成器22で生成される即値データの加算を行いメモリからデータを読み出す際に必要なアドレスデータを生成する。ALU40は数値演算や論理演算を行う。

【0086】また、該CPUは内部に各種バスや信号線を含んでいる。PA_BUS72やPB_BUS74はALU40の入力信号を伝送する機能等を有する。WW_BUS76はALU40の演算結果を取り出して汎用レジスタに伝送する機能等を有する。XA_BUS78は汎用レジスタ11やSP14から取り出したアドレスデータを伝送する機能等を有する。IA信号線82は、CPU内部の各部から外部のI_ADDR_BUS92へアドレスデータを伝送する。DA信号線84は、CPU内部の各部から外部のD_ADDR_BUS96へアドレスデータを伝送する。DIN信号線86は、CPU外部のD_DATA_BUS98からCPU内部の各部へデータを伝送する。DOU_T信号線88は、CPU内部の各部から外部のD_DATA_BUS98へデータを伝送する。IA入力切換83はIA信号線82へ出力される各種信号(PA_BUS72、WW_BUS76、PC12、PC+2)の切換を行う。DOU_T入力切換89はDOU_T信号線88へ出力される各種信号(PA_BUS72、WW_BUS76、PC12、PC+2)の切換を行う。

【0087】また、CPU10の前記各部は前記命令デコーダ20の出力する制御信号に基づき、命令の実行をおこなうもので、スタックポインタ専用命令群を、前記制御信号及び前記スタックポインタ専用レジスタの内容に基づき実行する前記実行手段としても機能する。

【0088】本CPU10は、16ビットの命令データバス(I_DATA_BUS)94、命令データアクセスのための命令アドレスバス(I_ADDR_BUS)92と、32ビットのデータバス(D_DATA_BUS)98と、データアクセスのためのデータアドレスバス(D_ADDR_BUS)96と、コントロール信号のための図示しないコントロールバスを介して外部と信号のやり取りを行う。

【0089】(2)本実施例のCPU有するレジスタセットの説明

次に本実施例のCPUが有するレジスタセットの概要について、必要な部分について説明する。

【0090】図2に本実施例のCPUの持つレジスタセットを示す。本実施例のCPUは、汎用レジスタ11を16本と、PC12、PSR13、SP14、図示しないALR(算術演算ローレジスタ)15、図示しないAHR(算術演算ハイレジスタ)16を含むレジスタセットを有している。

【0091】前記汎用レジスタ11は機能的に等価な32ビットのレジスタであり、R0からR15と名付けられている。該汎用レジスタ11はデータ演算時及びアドレ

ス計算時に使用される。

【0092】またPC12は、32ビット長のインクリメンタルカウンタであり、現在実行中の命令のアドレスであるプログラムカウンタを保持している。本文中で、レジスタ名を指すときはPCといい、PCに格納された値をさすときはプログラムカウンタという。

【0093】該PC12は、ロード命令等で直接アクセスすることは出来ない。call命令、int命令や割り込み、例外が発生すると、プログラムカウンタはPC12から読み出されスタックに退避される。この様に分岐命令が実行されると飛びさきアドレスがPCに設定される。条件分岐命令で分岐する場合も同様である。そして、ret命令やreti命令により、戻りさき命令アドレスがスタックより読み出され、PC12に復帰される。

【0094】PSR(プロセッサステータスレジスタ)13は、フラグが割り当てられている32ビットレジスタであり、CPUの現在の状態を保持している。int命令、割り込み、例外等が発生すると、それぞれの処理ルーチンに分岐する際に、その時のPSRの状態がスタックに退避される。逆にreti命令の実行で、退避されていた値がPSRに復帰される。

【0095】SP14は32ビットのスタックポインタ専用のレジスタで、スタックの先頭番地をさすスタックポインタが格納されている。本文中で、レジスタ名を指すときはSPといい、SPに格納された値をさすときはスタックポインタという。但し、スタックポインタは常にワードの境界を指しているため、前記スタックポインタの下位の2ビットは常に0である。

【0096】該スタックポインタは、本実施例で用意しているスタックポインタ専用命令群に含まれる各種命令の実行や、トラップの発生に伴い更新される。スタックポインタの更新を行うスタックポインタ専用命令としては、call命令やret命令等の他のルーチンへ分岐する命令、スタックポインタ移動命令、pushn命令、popn命令等がある。例えばcall命令が実行されると、まずスタックポインタがワードサイズ(4)だけデクリメント(-4)され、PC12がスタックに退避される。また、ret命令が実行されると、逆にスタックより戻りさきアドレスがPCにロードされ、スタックポインタはワードサイズ分インクリメント(+4)される。int命令が実行されたり、または、割り込み、例外等が発生したときには、スタックにPCやPSRの値が以下の手順で退避される。

【0097】1) SP=SP-4

2) スタックポインタで指されるスタックの先頭番地にPCを退避する。

【0098】3) SP=SP-4

4) スタックポインタで指されるスタックの先頭番地にPSRを退避する。

【0099】`reti`命令が実行されると、上記とは逆の処理を行ってCPUは以前の状態に戻る。このように`call`命令や`ret`命令や`int`命令が実行されると、該実行に基づきスタックポインタが更新される。各命令の詳細については後述する。

【0100】また、本実施例においてトラップとは、命令の実行に非同期に発生する割り込みと命令の実行により発生する例外を合わせたものをいう。トラップが発生すると、CPUはプログラムカウンタ(PC)とプロセッサステータスレジスタ(PSR)をスタックに退避し

特殊レジスタ名	特殊レジスタ番号	アセンブラの記述方法
プロセッサステータスレジスタ	0	%PSR
SP	1	%SP
算術演算ローレジスタ	2	%ALR
算術演算ハイレジスタ	3	%AHR

(3) スタック及びスタックポインタについての説明
スタックはメモリにもうけられた一時記憶領域で、連続した領域の下方からデータが棚状に積み重ねて記憶される。スタックポインタは、スタックメモリの一番上つまり最後に記憶されたデータのアドレスを示している。

【0105】スタックポインタの一般的な動作について図3を用いて説明する。

【0106】図3の100はメモリに設けられたスタック領域を表している。斜線部分102が最後に格納されたデータであるとする、SP14に格納されたスタックポインタは、該データのメモリアドレス1000を示している。なお、斜線部分102の下方領域104は既にデータが格納されている領域で、斜線部分102の上方領域106はこれからデータが格納される領域を示している。

【0107】スタックポインタは常にワードの境界を指しているため、スタックに情報を書き込む時はSP14の格納されたスタックポインタを4だけ上に動かして、該スタックポインタのさす場所に情報を格納する。またスタックに格納されている情報を取り出すときは、現在SP14が示すアドレスの情報をとりだして、SP14に格納されたスタックポインタを4だけ下に動かす。このようにスタックポインタは常に最後にスタックに格納された情報の格納アドレスを示している。

【0108】(4) スタックポインタ専用命令群についての説明

通常RISC方式のCPUでは汎用レジスタをスタックポインタとして使用するが、本実施例ではスタックポ

* 定する。

【0101】トラップ処理から元のルーチンに復帰するには、`reti`命令を使う。`reti`命令を実行すると、CPUはスタックからPSR、PCの順に読み出して、PSRを元の値に戻すとともに、戻りアドレスに分岐する。なお、例外にはデバッグ例外、アドレス不整列例外、オーバーフロー例外、ゼロ除算例外等がある。

【0102】ALR(算術演算ローレジスタ)15及びAHR(算術演算ハイレジスタ)16については説明を省略する。

【0103】CPUの持つ特殊レジスタPSR13、SP14、ALR15、AHR16はロード命令を使って汎用レジスタとの間でデータ転送を行うことが出来る。各レジスタは、特殊レジスタ番号を持っており、この番号を使ってアクセスされる。

【0104】

スタック専用のレジスタであるSP14を有しており、前記スタックポインタ専用命令群の操作対象となる。

【0109】前記スタックポインタ専用命令群は、スタックポインタ専用のレジスタであるSP14を暗黙のオペランドとし、SP14をもちいて操作をおこなう複数の命令の総称である。該スタックポインタ専用命令群は、SP相対ロード命令(`ld`等)、スタックポインタ移動命令(`add`、`sub`)、サブルーチンへ分岐する命令(`call`等)及びリターン命令(`ret`等)等の他のルーチンへ分岐する命令、連続プッシュ命令(`pushn`)、連続ポップ命令(`popn`)を含む。

【0110】これらにの命令に共通するのは、スタックポインタ専用命令であるため、オブジェクトコードにスタックポインタを特定するための情報が必要ないということである。従ってスタックポインタを用いる処理を短い命令長で効率よく記述することができるという効果も共通する。

【0111】また、これらの命令を用いるとメモリに設けられたスタックに記憶されている情報の処理を効率よく行うことができる。また、割り込み処理及びサブルーチンコール・リターンの処理を効率よく行うことが出来る。

【0112】ここで、サブルーチンがコールされる場合の前記スタックポインタ専用命令の使用例とメモリにもうけられたスタックの使用状態及びスタックポインタの状態について図4を用いて説明する。図4のメインプログラム500とサブルーチン520は、Cコンパイラが作成したオブジェクトコードで記述されたプログラムで

ある。540はメモリ上のスタックの状態を示している。502は汎用レジスタR0～R3を使用する処理が行われていることを示している。506はサブルーチンコール命令を示している。メインプログラム500でサブルーチンコール命令が実行されるサブルーチンコール命令前、即ち504に示す前の命令が実行された後スタックポインタ(SP)はメモリ上のスタック540上の①のアドレスを指していたとする。前記サブルーチンコール命令が実行されると、実行の制御はサブルーチン520にわたる。このとき本実施例では、前記スタックポインタ専用命令であるサブルーチンへ分岐する命令(call命令)が実行される。該命令が実行されると、スタックポインタ(SP)の値は自動的に-4だけインクリメントされ(図4の②SP参照)、該スタックポインタ②SPが指しているスタック上のエリア544にメインプログラムへのリターンアドレスが格納される。

【0113】そして、サブルーチン520側では、実行の最初にメインプログラムで使用されていた汎用レジスタR0～R3に格納されていた値をスタックに転送する処理を行う。524は、汎用レジスタR0～R3に格納されていた値をスタックに待避する処理を行うスタックポインタ専用命令である連続プッシュ命令(push)を示している。該命令が実行されると汎用レジスタR0～R3に格納されていた値が連続的にスタックに転送され、図4の550に示すように、スタック540に格納される。この処理の実行が終わる(524③)と、スタックポインタ(SP)は546を指している(③SP)。

【0114】次にサブルーチン520では、サブルーチンで使用するオート変数領域の確保を行う。526に示したadd命令は、スタックポインタ専用命令であるスタックポインタ移動命令であり、スタックポインタを上方に移動させサブルーチン520で使用するスタック領域を確保する。該命令が実行されるとスタックポインタ(SP)はXバイトだけ上方の位置548に移動し(④SP)、注2)に示すサブルーチンが使用するオート変数の領域が確保される。

【0115】528はサブルーチン520でオート変数と汎用レジスタR0～R3を用いた処理を行うことを示している。このときスタックポインタの位置は⑤SPを指しており、オート変数のロードは、スタックポインタ専用命令であるSP専用ロード命令をもちいて行われる。

【0116】529は、メモリ上に格納されたオート変数S1を汎用レジスタR1に転送する前記SPロード命令を示している。前記オート変数S1はスタックポインタ(⑤SP)からYバイトだけオフセット値を有する場所に格納されている。前述したようにサブルーチンの528の処理の最中にはスタックポインタは移動しないので、オート変数のメモリアドレスは、スタックポインタ

+オフセット値で特定され、前記SP専用ロード命令を用いて効率良く汎用レジスタとのやり取りを行うことができる。

【0117】また、サブルーチン520からメインプログラム500に戻る前には、スタックに待避されていた汎用レジスタR0～R3の値を汎用レジスタR0～R3に復帰させ、スタックポインタがメインルーチンへのリターンアドレスが格納されているエリア544を指すようにしなければならない。そのためにまず、526のスタックポインタ移動命令で移動させたスタックポインタをもとの位置に戻してやる。530に示したsub命令は、スタックポインタ専用命令であるスタックポインタ移動命令であり、スタックポインタを下方に移動させる。該命令が実行されると、スタックポインタは546に移動する(⑥SP参照)。次にスタックに格納されている情報550を汎用レジスタR0～R3に復旧する処理を行う。532は、スタックの情報を汎用レジスタR0～R3に転送する処理を行うスタックポインタ専用命令である連続ポップ命令(popn)を示している。該命令が実行されるとスタックに格納されていた値550が汎用レジスタR0～R3に連続的に転送され、図4の550に示すように、スタック540に格納される。この処理の実行が終わる(532⑦)と、スタックポインタは544を指している(⑦SP参照)。

【0118】534はリターン命令を示している。前記リターン命令が実行されると、実行の制御はメインプログラム500にわたる。このとき本実施例では、前記スタックポインタ専用命令であるリターン命令(ret命令)が実行される。該命令が実行されると、⑦SPが指すスタックエリアに格納されたメインプログラムへのリターンアドレスが示す命令に分岐する、即ちメインプログラム500の次の命令507に戻る。そしてスタックポインタの値は自動的に+4だけインクリメントされメインプログラム500が使用するスタックの先頭エリアに移動する(図4の⑧SP参照)。

【0119】以下前記各スタックポインタ専用命令についてそれぞれ命令の説明及び該命令を実行するための回路構成及び実行時の動き等について詳細に説明する。

【0120】(5)スタックポインタ相対ロード命令
前述したようにCコンパイラはオート変数の領域をスタックポインタに関連づけて記憶するオブジェクトコードを作成する。具体的には、CコンパイラはスタックポインタからオフセットがXのところにYバイトだけオート変数の領域を確保する仕様になっている。

【0121】図5は、メモリ上のオート変数をレジスタに転送する処理を説明するための図である。オート変数aはワードデータ、オート変数b、cはハーフワードデータ、オート変数d～gはバイトデータである。SPに格納されたスタックポインタはスタック上のオート変数が格納される領域600の先頭領域のアドレスである1

000を示している。オート変数aを格納するためのエリアとして前記スタックポインタをメモリアドレスとするスタック上のエリアが、オート変数b、cを格納するためのエリアとしてそれぞれスタックポインタ+2、スタックポインタ+4をメモリアドレスとするスタック上のエリアが、オート変数d~gを格納するためのエリアとしてそれぞれスタックポインタ+5、スタックポインタ+6、スタックポインタ+7、スタックポインタ+8をメモリアドレスとするスタック上のエリアが確保されている。そして所与の処理を実行する際には、スタックポインタ+オフセット値のメモリアドレスで特定されるオート変数と汎用レジスタ間でデータの転送処理が必要となる。

【0122】本実施例のCPUでは、このような転送処理を短いオブジェクトコードで記述し効率良く実行するために、スタックポインタ専用命令の一つであるSP相対ロード命令として、次に示す命令セットを用意している。

【0123】

```
ld. b  %Rd, [%sp+imm6]  ... (1)
ld. ub %Rd, [%sp+imm6] ... (2)
ld. h  %Rd, [%sp+imm7]  ... (3)
ld. uh %Rd, [%sp+imm7] ... (4)
ld. w  %Rd, [%sp+imm8]  ... (5)
ld. b  [%sp+imm6], %Rs   ... (6)
ld. h  [%sp+imm7], %Rs   ... (7)
ld. w  [%sp+imm8], %Rs   ... (8)
```

(1)~(8)は命令コードをアセンブラで記述したものである。(1)はスタックからバイトデータをサイン拡張してレジスタに転送する命令であり、(2)はスタックからバイトデータをゼロ拡張してレジスタに転送する命令であり、(3)はスタックからハーフワードデータをサイン拡張してレジスタに転送する命令であり、

(4)はスタックからハーフワードデータをゼロ拡張してレジスタに転送する命令であり、(5)はスタックからワードデータをレジスタに転送する命令であり、

(6)はレジスタからバイトデータをスタックに転送する命令であり、(7)はレジスタからハーフワードデータをスタックに転送する命令であり、(8)はレジスタからワードデータをスタックに転送する命令である。

【0124】[%sp+imm6]、[%sp+imm7]、[%sp+imm8]は、それぞれ即値オフセット情報を表しており、該即値オフセット情報に基づき実行時に生成されるオフセット値とSP14に格納されたスタックポインタの値を加算してメモリアドレスが生成される。

[%sp+imm6]はメモリ上のデータのデータサイズがバイトの時、[%sp+imm7]はメモリ上のデータのデータサイズがハーフワードの時、[%sp+imm8]はメモリ上のデータのデータサイズがワードの時の即値オフセット情報である。これらはいずれも、後述す

る図6(A)に示す用にオブジェクトコード上では6ビットの即値オフセット情報614として記述されている。しかし命令実行時には、imm7の場合(即ちデータサイズがハーフワードの場合)、即値オフセット情報614は1ビット左論理シフトしてスタックポインタに加算すべきオフセット値が生成される。また、imm8の場合(即ちデータサイズがワードの場合)、即値オフセット情報614は2ビット左論理シフトしてスタックポインタに加算すべきオフセット値が生成される。

【0125】ここにおいてスタックとはメモリに設けられた一時記憶領域をいい、前記メモリアドレスで、例えば図5に示すスタック上のオート変数(a~g)の位置を指定することができる。

【0126】図6(A)は、前記(1)~(8)のSP相対ロード命令のビットフィールド610である。図6(A)に示すようにSP相対ロード命令は、操作機能がメモリとレジスタ間のデータの転送であることを示すオペコード612(6ビット)、即値で指定された即値オフセット情報(6ビット)614と、転送対象となる汎用レジスタのレジスタ番号616(4ビット)を含み、16ビットのオブジェクトコードを有している。前記オペコード612はSP14を操作対象とするロード命令であることを示す共通のコードと前記データサイズ、サイン拡張及びゼロ拡張の別に応じて与えられる異なるコードとを含んでいる。従ってオペコードによりSP14に格納されたスタックポインタを操作対象とすることがわかるため、オブジェクトコードのオペランドにスタックポインタに関する情報を必要としない。即値オフセット情報614は転送対象となるデータのスタックポインタからのオフセット値を生成するための情報である。これは、前述したようにデータサイズを問わず6ビットで指定される。転送対象となるレジスタのアドレス616には、前記(1)~(5)の場合はスタックからよみだされたデータが格納されるレジスタのレジスタ番号が、前記(6)~(8)の場合はスタックに書き込まれるデータが格納されているレジスタのレジスタ番号が入っている。

【0127】図6(B)はスタックポインタとして汎用レジスタを用いた場合に使用する汎用レジスタを操作対象とするロード命令(以下汎用ロード命令という)のオブジェクトコードのビットフィールド620の例を示している。

【0128】図6(B)に示すように汎用ロード命令は、操作機能がメモリと汎用レジスタ間のデータの転送であることを示すオペコード622(6ビット)、即値で指定された即値オフセット情報624(6ビット)とスタックポインタとして使用する汎用レジスタを特定する第一のレジスタ番号626(4ビット)と、転送対象となる汎用レジスタを特定する第二のレジスタ番号628(4ビット)を含み、20ビットのオブジェクトコー

ドを有している。汎用マイクロコンピュータでは命令長は8ビット単位なので24ビット若しくは32ビット命令になる。

【0129】図6(A)(B)は、いずれもスタックポインタにオフセット値を加算して特定されるメモリアドレスとレジスタの間でのデータの転送処理を行う場合に使用する命令のオブジェクトコードを示しているが、同図に示すように、SP相対ロード命令は汎用ロード命令に比べて短いオブジェクトコードで記述することができる。

【0130】以下、スタックからレジスタにデータを転送する命令として(5)の命令(スタックからレジスタにワードデータを読み出す命令なので、以下ワードデータ読み出しのSP相対ロード命令という)、レジスタからスタックにデータを転送する命令として(8)の命令(レジスタからスタックにワードデータを取り込む命令なので、以下ワードデータ書き込みのSP相対ロード命令という)を例にとりこれらの命令を実行するための構成及び実行時の動作について説明する。

【0131】まず図1を用いてこれらの命令を実行するために必要なハードウェア構成について説明する。これらの命令は外部のメモリ(ROM)52よりI_DATA_BUS94を介して伝送され、CPU10の命令デコーダ20に入力される。該命令デコーダ20で、命令が解読され命令の実行に必要な図示しない各種信号が出力される。また前記即値生成器22は、データサイズに応じて前記即値オフセット情報614の左論理シフトを行い、必要に応じてサイン拡張及びゼロ拡張を行い実行に使用するオフセット値を生成し、PB_BUS74に出力する。SP14はスタックポインタを格納しており、この値はアドレス加算器30の入力に接続されたXA_BUS78に出力できる。アドレス加算器30のもう一方の入力は、即値生成器22の出力であるPB_BUS74に接続されている。アドレス加算器30の出力(ADDR)は、DA信号線84を介して外部のD_ADDR_BUS96に接続されている。

【0132】バスコントロールユニット(BCU)60は、CPUから出力される各種リクエスト信号(外部バスに出力された信号等)に基づき、スタックエリアを含むメモリ(RAM、ROM)50、52とのデータの入出力を制御し、READ、WRITE制御信号を出力する。

【0133】まずワードデータ読み出しのSP相対ロード命令の実行時の動作について説明する。

【0134】ワードデータ読み出しのSP相対ロード命令が実行されると、SP14に格納されたスタックポインタの値と、前記即値オフセット情報614に基づき即値生成器22が生成したオフセット値が加算され、メモリ読み出し用のメモリアドレスが生成される。そして該メモリアドレスに基づきメモリ上の情報が読み出され、

オブジェクトコードの前記レジスタ番号616で特定される汎用レジスタに転送される。

【0135】図7はワードデータ読み出しのSP相対ロード命令の動作を説明するためのフローチャート図である。

【0136】前記命令の実行の最初にSP14に格納されているスタックポインタがXA_BUS78に出力される(ステップ210)。また、即値生成器22が即値オフセット情報から生成したオフセット値immがPB_BUS74に出力される(ステップ212)。アドレス加算器30は、前記XA_BUS78上の値と前記PB_BUS74上の値を加算し、結果(ADDR)であるメモリの読み出しアドレスをDA信号線84を介してD_ADDR_BUS96に出力する(ステップ214、ステップ216)。そしてCPUからBCU60へのデータ読み出しリクエスト信号がアクティブになり、外部メモリのリードサイクルを実行する(ステップ218)。即ち前記BCU60は該リクエスト信号に基づき、前記読み出しアドレスをメモリアドレスとしてメモリからデータが読み出され、D_DATA_BUS98に出力されるよう制御する。D_DATA_BUS98上のデータはDIN信号線86を介してWW_BUS76に出力される(ステップ220)。そしてWW_BUS76上の値は命令コードの転送対象となるレジスタ(Rs/Rd)のアドレス(4ビット)616で指定されたレジスタ番号を有するレジスタ(%Rd)に格納される(ステップ222)。

【0137】次にワードデータ書き込みのSP相対ロード命令の実行時の動作について説明する。

【0138】ワードデータ書き込みのSP相対ロード命令が実行されると、SP14に格納されたスタックポインタの値と、前記即値オフセット情報614に基づき即値生成器22が生成したオフセット値が加算され、メモリ書き込み用のメモリアドレスが生成される。そしてオブジェクトコードの前記レジスタ番号616で特定される汎用レジスタに格納されている情報が、該メモリアドレスに基づき特定されるメモリ上のエリアに転送される。

【0139】図8はワードデータ書き込みのSP相対ロード命令の動作を説明するためのフローチャート図である。

【0140】前記命令の実行の最初にSP14に格納されているスタックポインタがXA_BUS78に出力される(ステップ230)。また、即値生成器22が即値オフセット情報から生成したオフセット値immがPB_BUS74に出力される(ステップ232)。アドレス加算器30は、前記XA_BUS78上の値と前記PB_BUS74上の値を加算し、結果(ADDR)であるメモリへの書き込みアドレスをDA信号線84を介してD_ADDR_BUS96に出力する(ステップ23

4、ステップ236)。また、命令コードの転送対象となるレジスタ(Rs/Rd)のアドレス(4ビット)616で指定されたレジスタ番号を有するレジスタ(%Rd)に格納されているデータがPA_BUS72に出力される(ステップ238)。PA_BUS72上のデータはDOUT信号線88を介して、D_DATA_BUS98に出力される(ステップ240)。そしてCPUからBCU60へのデータ書き込みリクエスト信号がアクティブとなり、外部メモリのライトサイクルを実行する(ステップ242)。即ち前記BCU60は前記リクエスト信号に基づき、前記書き込みアドレスをメモリアドレスとして、D_DATA_BUS98で転送されてきたデータをメモリ50に書き込む動作を制御する。

【0141】(6)スタックポインタ移動命令

図9(A)～(F)は、複数のルーチンにわたってプログラムが実行される場合の各ルーチンによるメモリ上のスタックの使用状態及びスタックポインタの状態を説明するための図である。

【0142】図9(A)に示すMAINルーチン210の所与の処理aの実行時におけるメモリ上のスタック領域の状態及びスタックポインタ(SP14に格納されている値)の状態を図9(D)に示している。222はMAINルーチン210で使用するためのスタック領域を示しており、スタックポインタは222の先頭アドレス232を示している。

【0143】図9(B)のSUB1ルーチン212は、前記MAINルーチン210から呼ばれて実行されるサブルーチンである。該SUB1ルーチン212の所与の処理b213の実行時におけるメモリ上のスタック領域の状態及びスタックポインタ(SPに格納されている値)の状態を図9(E)に示している。224はSUB1ルーチン212で使用するためのスタック領域を示しており、スタックポインタは224の先頭アドレス234を示している。

【0144】図9(C)のSUB2ルーチン214は、前記SUB1ルーチン210から呼ばれて実行されるサブルーチンである。該SUB2ルーチン214の所与の処理c215の実行時におけるメモリ上のスタック領域の状態及びスタックポインタ(SPに格納されている値)の状態を図9(F)に示している。226はSUB2ルーチン214で使用するためのスタック領域を示しており、スタックポインタは226の先頭アドレス236を示している。

【0145】このように複数のサブルーチンにわたって実行が行われる場合、各ルーチンで使用するスタック領域が移動するため、それに伴いスタックポインタの値を、各ルーチンで使用するスタック領域の先頭に移動することが行われる。

【0146】本実施例のCPUでは、このようなスタックポインタの移動処理を短いオブジェクトコードで記述

し効率良く実行するために、スタックポインタ専用命令の一つであるスタックポインタ移動命令として、次に示す命令セットを用意している。

【0147】

add %sp, imm12 … (9)

sub %sp, imm12 … (10)

(9)(10)は命令コードをアセンブラで記述したものである。(9)はSP14に格納されたスタックポインタに対する即値加算命令であり、(10)はSP14に格納されたスタックポインタに対する即値減算命令である。imm12は、命令のオブジェクトコードに含まれた10ビットの即値を2ビット左方向にシフトした後、ゼロ拡張されて32ビットデータとなり、SP14に格納されたスタックポインタとの演算に用いられる。

【0148】図10(A)は、前記(9)(10)のスタックポインタ移動命令のビットフィールド630である。図10(A)に示すようにスタックポインタ移動命令は、操作機能がSP14に格納されたスタックポインタへの移動情報の加算及び減算であることを示すオペコード632(6ビット)、即値で指定された即値移動情報634(10ビット)を含み、16ビットのオブジェクトコードを有している。前記オペコード632はSP14を操作対象とするスタックポインタ移動命令であることを示す共通のコードと加算及び減算の別に応じて与えられる異なるコードとを含んでいる。従ってオペコードによりSP14に格納されたスタックポインタを操作対象とすることがわかるため、オブジェクトコードのオペランドにスタックポインタに関する情報を必要としない。即値移動情報634はスタックポインタに加算又は減算を行うオフセット値を生成するための情報である。

【0149】図10(B)はスタックポインタとして汎用レジスタを用いた場合に使用する加算及び減算命令(以下汎用演算命令という)のオブジェクトコードのビットフィールド640の例を示している。

【0150】図10(B)に示すように汎用演算命令は、操作機能が汎用レジスタの値への即値演算情報の加算及び減算であることを示すオペコード642(6ビット)、即値で指定された即値演算情報644(10ビット)と操作対象となる汎用レジスタを特定するレジスタ番号646(4ビット)とを含み、20ビットのオブジェクトコードを有している。汎用マイクロコンピュータでは命令長は8ビット単位なので24ビット若しくは32ビット命令になる。

【0151】図10(A)(B)は、いずれもスタックポインタに即値を加算及び減算する処理を行う場合に使用する命令のオブジェクトコードであるが、同図に示すように、スタックポインタ移動命令は汎用演算命令に比べて短いオブジェクトコードで記述することができる。

【0152】以下、(9)のSPに対する即値加算命令(以下加算スタックポインタ移動命令という)と、(1

0)のSPに対する即値減算命令(以下減算スタックポインタ移動命令という)を実行するための構成及び実行時の動作について説明する。

【0153】まず図1を用いてこれらの命令を実行するために必要なハードウェア構成について説明する。これらの命令は外部のメモリ(ROM)52よりI_DAT_A_BUS94を介して伝送され、CPU10の命令デコーダ20に入力される。該命令デコーダ20で、命令が解読され命令の実行に必要な図示しない各種信号が出力される。また前記即値生成器22は、前記即値移動情報634の10ビットを2ビット左論理シフトし、ゼロ拡張してPA_BUS72に出力する。SP14はスタックポインタを格納しており、この値はXA_BUS78に出力される。XA_BUS78はALU40の入力となるPB_BUS74に接続されている。ALU40のもう一方の入力は、即値生成器22の出力であるPA_BUS72に接続されている。ALU40の出力は、WW_BUS76に接続されている。WW_BUS76はSPの入力に接続されている。

【0154】まず加算スタックポインタ移動命令の実行時の動作について説明する。

【0155】加算スタックポインタ移動命令が実行されると、SP14に格納されたスタックポインタの値と、前記即値移動情報634に基づき即値生成器22が生成した移動即値が加算されて新たなスタックポインタが生成され、その値がSP14に格納される。

【0156】図11は加算スタックポインタ移動命令の動作を説明するためのフローチャート図である。

【0157】前記命令の実行の最初にSP14に格納されているスタックポインタがXA_BUS78に出力される(ステップ250)。そして前記XA_BUS78上のデータはPB_BUS74に出力される(ステップ252)。また、即値生成器22が即値移動情報に基づき生成した移動即値immがPA_BUS72に出力される(ステップ254)。ALU40は、前記PB_BUS74上の値と前記PA_BUS72上の値を加算し、結果をWW_BUS76に出力する(ステップ256)。そして、WW_BUS76上の値がSP14に入力される(ステップ258)。

【0158】次に減算スタックポインタ移動命令の実行時の動作について説明する。

【0159】減算スタックポインタ移動命令が実行されると、SP14に格納されたスタックポインタの値から、前記即値移動情報634に基づき即値生成器22が生成した移動即値が減算されて新たなスタックポインタが生成され、その値がSP14に格納される。

【0160】図12は減算スタックポインタ移動命令の動作を説明するためのフローチャート図である。

【0161】前記命令の実行の最初にSP14に格納されているスタックポインタがXA_BUS78に出力さ

れる(ステップ260)。そして前記XA_BUS78上のデータはPB_BUS74に出力される(ステップ262)。また、即値生成器22が即値移動情報に基づき生成した移動即値immがPA_BUS72に出力される(ステップ264)。ALU40は、前記PB_BUS74上の値から前記PA_BUS72上の値を減算し、結果をWW_BUS76に出力する(ステップ266)。そして、WW_BUS76上の値がSP14に入力される(ステップ268)。

10 【0162】(7)分岐命令

図13は、call命令とret命令によるプログラムの実行の制御を説明するための図である。図13に示すように、MAINルーチン300において、サブルーチンSUB310へ分岐するためのcall命令が実行されると(302)、制御がサブルーチンSUB310に渡る。サブルーチンの最後には、ret命令(312)が書かれており、この命令が実行されると、MAINルーチン300の前記call命令(302)の次の命令(304)に戻る。従って、図13に示す場合①②③の順でプログラムが実行されることになる。このようにサブルーチンSUB310での実行が終わるとMAINルーチン300に戻り、前記call命令(302)の次の命令(304)から実行するため、サブルーチンSUB310に分岐する際、どこかに戻り先のアドレスを記憶しておくことが必要となる。このため、call命令等のサブルーチンへ分岐する分岐命令実行の際には、図4で説明したように戻り先アドレスをスタックに待避する処理が行われ、ret命令等のサブルーチンから戻る分岐命令の実行の際には前記スタックから戻り先アドレスをプログラムカウンタに戻す処理(以下プログラムカウンタの待避及び復旧の処理という)が行われる。

【0163】従来のRISC方式のCPUでは、前記プログラムカウンタの待避及び復旧の処理をソフトウェア的に実現していたため、call命令等の分岐命令を実行する際には、これらの処理を実行するためオブジェクトコード(アセンブラ命令)も必要であった。例えばcall命令を実行する際には、スタックポインタをワードサイズ(4)だけデクリメント(-4)して、プログラムカウンタの値に基づきcall命令の次の命令のアドレスをスタックに格納するためのオブジェクトコード(アセンブラ命令)が必要であった。

【0164】しかし本実施例のCPUは、前記call命令やret命令が実行されると、前記プログラムカウンタの待避及び復旧の処理も一緒に行うようなハードウェア構成を有している。従って、前記call命令やret命令と別に、前記プログラムカウンタの待避及び復旧の処理を記述するオブジェクトコード(アセンブラ命令)を必要としない。本実施例のCPUでは、このような前記プログラムカウンタの待避及び復旧の処理を一命令で実行するために、スタックポインタ専用命令の一つ

である分岐命令として次に示す命令セットを用意している。

```

【0165】call sign9      ... (11)
call %Rb                ... (12)
ret                     ... (13)
reti                   ... (14)
ret d                   ... (15)
int imm2                ... (16)
brk                     ... (17)

```

(11)～(17)は命令コードをアセンブラで記述したものである。(11)はPC相対サブルーチンコール命令で、プログラムカウンタPCをベースアドレスとして、オペランドで指定されたディスプレースメント情報sign9に基づき分岐先アドレスを相対的に指定して分岐するコール命令である。(12)はレジスタ間接サブルーチンコール命令であり、オペランドで指定されたレジスタに格納されている分岐先アドレスに分岐するコール命令である。(13)はサブルーチンからのリターン命令である。(14)は割り込み又は例外処理ルーチンからのリターン命令である。(15)はデバッグ処理ルーチンからのリターン命令である。(16)はソフトウェア割り込み命令である。(17)はソフトウェア・デバッグ割り込み命令である。

【0166】図14は、前記(11)のPC相対サブルーチンコール命令のビットフィールド650である。図14に示すようにPC相対サブルーチンコール命令は、操作機能がプログラムカウンタをベースアドレスとして、分岐先アドレスを相対的に指定してサブルーチンへ分岐するコール命令であることを示すオペコード652(8ビット)、即値で指定されたディスプレースメント情報sign9(8ビット)654とを含み、16ビットのオブジェクトコードを有している。前記8ビットの即値は、実行時に1ビット左に論理シフトされた後、サイン拡張される。

【0167】本実施例のCPUでは、call命令実行には図14に示すオブジェクトコードのみで、プログラムカウンタのスタックへの待避も実行することが出来る。

【0168】以下、サブルーチンへ分岐する命令として(11)のPC相対サブルーチンコール命令、サブルーチンからリターンする命令として(13)のリターン命令を例にとりこれらの命令を実行するための構成及び実行時の動作について説明する。

【0169】まず図1を用いてこれらの命令を実行するために必要なハードウェア構成について説明する。これらの命令は外部のメモリ(ROM)52よりI_DATA_BUS94を介して伝送され、CPU10の命令デコード20に入力される。該命令デコード20で、命令が解釈され命令の実行に必要な図示しない各種信号が出力される。また前記即値生成器22は、前記ディスプレ

ースメント情報654を1ビット左に論理シフトした後サイン拡張して32ビットの即値ディスプレースメントimmを生成し、PB_BUS74に出力する。SP14はスタックポインタを格納しており、この値はアドレス加算器30の入力に接続されたXA_BUS78に出力できる。アドレス加算器30のもう一方の入力は、即値生成器22の出力であるPB_BUS74に接続されている。アドレス加算器30の出力(ADDR)は、I_A信号線82を介して外部のI_ADDR_BUS92に接続されている。

【0170】また、I_ADDR_BUS92及びI_DATA_BUS94は命令のオブジェクトコードが格納されたROM52に接続されており、バスコントロールユニット(BCU)60は、CPUから出力される各種リクエスト信号(外部バスに出力された信号等)に基づき、メモリ(ROM)52から前記命令のオブジェクトコードを読み出すREAD制御信号を出力する。

【0171】まずPC相対サブルーチンコール命令の実行時の動作について説明する。

【0172】PC相対サブルーチンコール命令が実行されると、図4で説明したように、PC12に格納されたプログラムカウンタの値がスタックに待避され、SP14に格納されたスタックポインタの値がワードサイズ(4)だけデクリメントされる。そしてPC12にプログラムカウンタと前記32ビットの即値ディスプレースメントを加算して得られた分岐先アドレスがセットされる。

【0173】図15はPC相対サブルーチンコール命令の動作を説明するためのフローチャート図である。

【0174】前記命令の実行の最初にSP14に格納されているスタックポインタがXA_BUS78に出力される(ステップ270)。これはアドレス加算器の一方の入力となり、即値生成器22が生成したconstantデータ(-4)がアドレス加算器30の他方の入力となる。そしてアドレス加算器30で前記XA_BUS78上のスタックポインタの値と-4を加算して戻りアドレスを格納するスタックへの書き込みアドレス生成され、WW_BUS76に出力される。また、前記書き込みアドレスはDA信号線84を介してD_ADDR_BUS96に出力される。(ステップ272)。また、PCインクリメント44ではPC12に格納されたプログラムカウンタの値が+2されて戻りアドレスが生成されて、DO_UT信号線88を介してD_DATA_BUS98に出力される(ステップ274)。そしてCPUからBCU60へのデータ書き出しリクエスト信号がアクティブになり、外部メモリのライトサイクルを実行する(ステップ276)。即ち前記BCU60は該リクエスト信号に基づき、前記書き込みアドレスをメモリアドレスとしてメモリに設けられたスタックに前記戻りアドレスが格納される。

【0175】そして、WW_BUS76上の値がSP14に出力される(ステップ278)。すなわちスタックポインタの値が-4した値に更新される。

【0176】次にPC12に格納されているプログラムカウンタがXA_BUS78に出力される(ステップ280)。また、前記即値生成器22は、命令のオブジェクトコードに含まれている前記ディスプレースメント情報654を1ビット左に論理シフトした後サイン拡張して32ビットの即値ディスプレースメントimmを生成し、PB_BUS74に出力する。前記アドレス加算器30はXA_BUS78上のプログラムカウンタとPB_BUS74上の即値ディスプレースメントimmを加算し分岐アドレス(ADDR)を生成し、IA信号線82を介して、I_ADDR_BUS92に出力する(ステップ282)。そしてCPUからBCU60への命令読み出しリクエスト信号がアクティブとなり、外部メモリ(ROM)52のリードサイクルを実行し、分岐先の命令のオブジェクトコードを読み出す(ステップ284)。

【0177】次にret命令の実行時の動作について説明する。

【0178】ret命令が実行されると、図4で説明したように、スタックに待避されていたプログラムカウンタの値がPC12に復帰され、SP14に格納されたスタックポインタの値がワードサイズ(4)だけインクリメントされる。

【0179】図16はret命令の動作を説明するためのフローチャート図である。

【0180】前記命令の実行前には、SP14に格納されたスタックポインタは呼ばれたルーチンへの戻り先アドレスが格納されたスタックのアドレスをさしている。

【0181】前記命令の実行の最初にSP14に格納されているスタックポインタがXA_BUS78に出力される(ステップ290)。そしてXA_BUS78上のスタックポインタはD_ADDR_BUS96に出力される(ステップ292)。そしてCPUからBCU60へのデータ読み込みリクエスト信号がアクティブになり、外部メモリのリードサイクルを実行する。即ち前記BCU60は該リクエスト信号に基づき、前記スタックポインタをメモリアドレスとしてメモリに設けられたスタックから前記戻りアドレスを読み出す。メモリ(RAM)50から読み出された前記戻り先アドレスはD_DATA_BUS94からDIN信号線86を介してCPU内部にとりこまれ、さらにDIN信号線からIA信号線82を介してI_ADDR_BUS92に出力される(ステップ294)。そしてCPUからBCU60への命令読み出しリクエスト信号がアクティブとなり、外部メモリ(ROM)52のリードサイクルを実行し、戻り先アドレスの命令のオブジェクトコードを読み出す(ステップ296)。

【0182】そして、XA_BUS78上のスタックポインタの値は、前記アドレス加算器30の一方の入力となる。また即値生成器22が生成したconstantデータ(+4)がアドレス加算器30の他方の入力となる。そしてアドレス加算器30で前記XA_BUS78上のスタックポインタの値と+4を加算して、戻り先のルーチンが確保したスタック領域の先頭エリアのアドレスが生成され、WW_BUS76に出力される(ステップ298)。そして、WW_BUS76上の前記アドレス(戻り先のルーチンが確保したスタック領域の先頭エリアのアドレス)がSP14に出力される(ステップ300)。

【0183】(8)連続プッシュ命令(pushn)、連続ポップ命令(popn)の説明

前述したように、最近の特にRISC方式のCPUは、性能を高めるため内部に多くの汎用レジスタを持ち、メモリにアクセスすることなくCPU内部で高速に多くの処理をおこなうよう構成されている。本実施例でも内部に16本の汎用レジスタをもち、処理の高速化を図っている。しかし、このように内部レジスタを多く持つと、割り込み処理やサブルーチンコールの時のレジスタ退避と復旧の処理の際、退避すべきレジスタ数が多くなる。

【0184】このようなレジスタの待避や復旧を行う場合、従来は、番地部で指定した内容をスタックに格納するpush命令や、スタックの内容をレジスタに取り出すpop命令を用いていた。ここで一般的なpush命令及びpop命令時の動きを説明する。

【0185】図17(A)(B)はpush命令実行時の動きを模式的に示した図であり、図18(A)(B)はpop命令実行時の動きを模式的に示した図である。図17(A)(B)及び図18(A)(B)を用いて、複数の汎用レジスタとスタック間でデータの転送を行う場合の動きを説明する。

【0186】図17(A)は、汎用レジスタR1の内容をスタックに書き出す命令である'push R1'が実行された場合の動きを示している。該命令の実行時には、SP14の内容は現在の値から4がひかれた値に更新される(図17(A)に示すように1000が996に更新される)。そして、更新されたSP14のスタックポインタが示すメモリアドレスである996に汎用レジスタR1の内容aが書き込まれる。

【0187】図17(B)は、さらに汎用レジスタR2の内容をスタックに書き出す命令である'push R2'が実行された場合の動きを示している。該命令の実行時には、SP14の内容は現在の値から4がひかれた値に更新される(図17(B)に示すように996が992に更新される)。そして、更新されたSP14のスタックポインタが示すメモリアドレスである992に汎用レジスタR2の内容bが書き込まれる。

50 【0188】図18(A)は、スタックの内容を汎用レ

レジスタR2に取り出す命令である'pop R2'が実行された場合の動きを示している。該命令の実行時には、SP14のスタックポインタが示しているメモリアドレス992に格納されている内容bがとりだされて汎用レジスタR2に格納される。そしてSP14の内容は現在の値に4が足された値に更新される(図18(A)示すように実行前992であったのが実行後996に更新される)。

【0189】図18(B)は、更にスタックの内容を汎用レジスタR1に取り出す命令である'pop R1'が実行された場合の動きを示している。該命令の実行時には、SP14のスタックポインタが示しているメモリアドレス996に格納されている内容aがとりだされて汎用レジスタR1に格納される。そしてSP14の内容は現在の値に4が足された値に更新される(図18

(B)に示すように実行前996であったのが実行後1000に更新される)。

【0190】このように従来は、複数の汎用レジスタとスタックでデータの転送を行う場合、push命令やpop命令を複数回繰り返して実行することが必要であった。push命令やpop命令は1回の命令で1本のレジスタしか操作することができなかったからである。

【0191】従って多数のレジスタに対してスタックへの待避やスタックからの復旧処理を行う場合、命令数の増加によりオブジェクトコードのサイズの増大を招いていた。また、プログラムの実行ステップも多くなり、プログラムの実行時間や処理動作の遅延を招いていた。

【0192】そこで本実施例のCPUでは、次に示す命令セットを用意している。

【0193】pushn %Rs ... (18)
popn %Rd ... (19)

(18)は連続プッシュ命令のアセンブラの記述を示しており、%RsからR0までのn個(nは1から16の自然数)の汎用レジスタの内容を連続的にスタックへプッシュする命令である。(19)は連続ポップ命令のアセンブラの記述を示しており、スタックからn個(nは1から16の自然数)のワードデータを連続的に%RdからR0までの汎用レジスタへポップする命令である。pushn、popn命令は、いずれもオペランドとオペランドからなる。%Rsはpushn命令のオペランドでレジスタ%RsからR0までをスタックに書き込む場合のRsのレジスタ番号を示している。%Rdはpopn命令のオペランドでR0から%Rd迄のレジスタにスタックからデータを持ってくる場合のRdのレジスタ番号を示している。

【0194】図19にpushn、popn命令のビットマップを示す。下位の4ビットのフィールドに%Rsまたは%Rdを示すコードが入り、16本ある汎用レジスタの任意のレジスタが指定できる。特殊レジスタとスタックの間でデータの転送をするときは、汎用レジスタ

を介して行う。

【0195】図20は連続プッシュ命令(pushn)、連続ポップ命令(popn)を実行するためのハードウェア構成を説明するためのブロック図である。図1から連続プッシュ命令(pushn)及び連続ポップ命令(popn)の説明に必要な部分を取り出して、説明に必要な部分を追加した構成になっている。図1と同一部分を指すものについて同一の番号を付している。図中11はR0からR15の16本の汎用レジスタであ

る。汎用レジスタ11はデータバス(D_DATA_BUS)98とデータの入出力を行う。またレジスタを選択するレジスタ選択アドレス信号54はコントロール回路ブロック45から来る。14はSPでスタックポインタが格納されている。SP14の値はアドレスバス(D_ADDR_BUS)96と32ビットのアドレス加算器30の入力に接続する内部アドレスバス(XA_BUS)78に出力できる。アドレス加算器30のもう一方の入力はコントロール回路ブロック45からの出力されるオフセット信号24に接続されている。アドレス加算器30の出力はラッチ(Add_LT)32でラッチされ、さらにXA_BUS78またはWW_BUS76に出力される。WW_BUS76はSP14の入力に接続されている。コントロール回路ブロック45の中には4ビットのカウンタ(countx)46があり、転送するレジスタ数をカウントする。又、コントロール回路ブロック45中には図20では示されていないがインストラクションレジスタがあり、pushn、popn命令のオペランド%Rs、%Rdを保持するとともに、インストラクションデコードによって各命令に応じた制御信号を出力する。図中60はバスコントロールユニット(BCU)で外部のスタックエリアを含むメモリ(RAM)50とのデータの入出力を制御し、READ、WRITE制御信号を出力する。

【0196】まず連続プッシュ命令(pushn)の実行時の動作について説明する。

【0197】連続プッシュ命令(pushn)が実行されると、図4で説明したように、%Rsのレジスタ番号の汎用レジスタから汎用レジスタR0までの汎用レジスタに格納された内容が連続的にスタックにプッシュされる。

【0198】図21はpushn命令の動作を説明するためのフローチャート図である。

【0199】pushn命令の実行の最初にオフセット信号24のoffsetは-4になる。カウンタ(countx)46はゼロにクリアされ、SP14に格納されたスタックポインタ値がXA_BUS78に出力される(ステップ100)。

【0200】次にアドレス加算器30はXA_BUS78上の値と-4を加算し結果をラッチ(Add_LT)32に入れる(ステップ101)。

【0201】ラッチ (Add_LT) 32の値はアドレスバスD_ADDR_BUS96に出力される。コントロール回路ブロック45ではインストラクションレジスタの下位4ビットに保持された%Rsとカウンタ (countx) 46の差を計算し結果をレジスタ選択アドレス信号54に出力する。54によって選択されたレジスタをデータバス (D_DATA_BUS) 98にのせ、外部メモリ (RAM) 50の書き込み動作をする (ステップ102)。

【0202】ステップ103では、カウンタ (countx) 46と%Rsを比較する。等しいときはレジスタの外部メモリへの書き込みは完了しており、ラッチ (Add_LT) 32の値をWW_BUS76を通じてSP14に書き込んでpushnの実行を終わる (ステップ104)。

【0203】等しくないときはカウンタ (countx) をプラス1し、ラッチ (Add_LT) 32の値をXA_BUS78に出力し、ステップ101以降の処理を繰り返す (ステップ105)。

【0204】次に連続ポップ命令 (popn) の実行時の動作について説明する。

【0205】連続ポップ命令 (popn) が実行されると、図4で説明したように、スタックの内容が汎用レジスタR0から%Rdのレジスタ番号の汎用レジスタに連続的にプッシュされる。

【0206】図22は、popn命令の動作を説明するためのフローチャート図である。

【0207】popn命令の最初にオフセット信号24は+4になる。カウンタ (countx) 46はゼロクリアされ、SP14のスタックポインタの値がXA_BUS78とアドレスバス (D_ADDR_BUS) 96に出力される (ステップ110)。

【0208】次にアドレス加算器30はXA_BUS78上の値と+4を加算し結果をラッチ (Add_LT) 32に入れる (ステップ111)。

【0209】次に外部メモリのリードサイクルを実行する。リードされたデータはデータバス (D_DATA_BUS) 98を通じて汎用レジスタ11に書き込まれる。このときコントロール回路ブロック45ではカウンタ (countx) 46をレジスタ選択アドレス信号54に出力する (ステップ112)。

【0210】ステップ113では、カウンタ (countx) 46と%Rsを比較する。等しいときはレジスタの外部メモリへの書き込みは完了しており、ラッチ (Add_LT) 32の値をWW_BUS76を通じてSP14に書き込んでpopnの実行を終わる (ステップ114)。

【0211】等しくないときはカウンタ (countx) をプラス1し、ラッチ (Add_LT) 32の値をXA_BUS78とアドレスバス (D_ADDR_BU 50

S) 96に出力し、ステップ111以降の処理を繰り返す (ステップ115)。

【0212】このように 'pushn %Rs' で%RsからR0までのレジスタをスタックに書き込み、また 'popn %Rd' でスタックから必要な個数のデータをR0から%Rdまでのレジスタに戻すことが出来る。たとえば、'pushn %R3' の実行によりレジスタR3からR0までを一命令でプッシュできる。

【0213】ここにおいてレジスタの使い方に一つの制約を加えることで、更に有効な効果が発生する。即ち、レジスタの待避や復旧は割り込み処理やサブルーチンコール時等のプログラムが他のルーチンに分岐するときに特に必要になるが、このとき他の割り込みルーチンまたはサブルーチン等の呼び出されるルーチンではレジスタをR0から順に使って行くことが好ましい。このようにすると、R0からRdまたはRsに退避の必要のないレジスタが含まれず、pushn命令及びpopn命令を用いて、効率的にレジスタの待避又は復帰を行うことができる。本実施例ではレジスタはどれも同じ機能を持っており、レジスタの使い方と命令には何の制約も無いので、このような制約は何の問題もなく満たすことが出来る。

【0214】従って本実施例のpushn命令及びpopn命令を用いることにより、レジスタからメモリ中のスタックへの退避、またスタックからレジスタへのデータの復旧がpushn、popnの各々命令で実行できる。このためオブジェクトコードサイズやプログラム実行ステップを最少にし、かつ実行サイクルを一回の命令フェッチと必要な回数のデータの転送だけで済ませ、最少のサイクルで行うことが出来る。これにより割り込み処理ルーチンやサブルーチンの処理も高速化を図ることができる。

【0215】一方、これを実現する構成要素はカウント手段と簡単なシーケンス制御のみであり、少ないゲート数で実現出来、ワンチップのマイクロコンピュータに適したものである。

【0216】また、本実施例ではスタックポインタ専用のレジスタSP14を用いた場合の連続プッシュ命令 (pushn)、連続ポップ命令 (popn) を実行させるための構成について説明したが、スタックポインタとして汎用レジスタを使用する場合にも適用可能である。

【0217】(実施例2) 図23は、本実施例のマイクロコンピュータのハードウェアブロック図である。

【0218】該マイクロコンピュータ2は、32ビットマイクロコントローラであり、CPU10とROM52とRAM50、高周波発振回路910、低周波発振回路920、リセット回路930、プリスケラ940、16ビットプログラマブルタイマ950、8ビットプログラマブルタイマ960、クロックタイマ970、インテ

リジェントDMA980、高速DMA990、割り込みコントローラ800、シリアルインターフェース810、バスコントロールユニット(BCU)60、A/D変換器830、D/A変換器840、入力ポート850、出力ポート860、I/Oポート870、及びそれらを接続する各種バス92、94、96、98、各種ピン90等を含む。

【0219】前記CPU10は、スタックポインタ専用レジスタであるSPを有し、前述した各種のスタックポインタ専用命令の解釈、実行を行う。該CPU10は、

前述した、実施例1の構成を有しており、前記解釈手段、前記実行手段として機能する。

【0220】従って本実施例のマイクロコンピュータは、スタックポインタを取り扱う処理を、短い命令長で効率よく記憶し、実行することができる。

【0221】また、レジスタ退避やレジスタ復旧の処理を効率よく記憶し、割り込み処理及びサブルーチンコール・リターン処理を高速に行うことができる。

【0222】本発明のマイクロコンピュータは例えばプリンター等のパソコン周辺機器や、携帯機器等の各種の電子機器に適用可能である。この様にすると、簡単な構成でメモリの使用効率がよく高速に処理の行える情報処理回路を内蔵することができるため、安価で高機能な電子機器を提供することが出来る。

【0223】なお本発明は、上記実施例で説明したものに限らず、種々の変形実施が可能である。

【0224】

【図面の簡単な説明】

【図1】本実施例のCPUの回路構成の概略を説明するための図である。

【図2】本実施例のCPUの持つレジスタセットを示す。

【図3】スタックポインタの一般的な動作について説明するための図である。

【図4】スタックポインタ専用命令の使用例とメモリに与えられたスタックの使用状態及びスタックポインタの状態について説明するための図である。

【図5】メモリ上のオート変数をレジスタに転送する処理を説明するための図である。

【図6】図6(A)(B)は、SP相対ロード命令及び汎用ロード命令のビットフィールドを示した図である。

【図7】ワードデータ読み出しのSP相対ロード命令の動作を説明するためのフローチャート図である。

【図8】ワードデータ書き込みのSP相対ロード命令の動作を説明するためのフローチャート図である。

【図9】図9(A)～(F)は、複数のルーチンにわたってプログラムが実行される場合の各ルーチンによるメモリ上のスタックの使用状態及びスタックポインタの状態を説明するための図である。

【図10】図10(A)(B)は、スタックポインタ移

動命令及び汎用演算命令のビットフィールドを示した図である。

【図11】加算スタックポインタ移動命令の動作を説明するためのフローチャート図である。

【図12】減算スタックポインタ移動命令の動作を説明するためのフローチャート図である。

【図13】call命令とret命令によるプログラムの実行の制御を説明するための図である。

【図14】PC相対サブルーチンコール命令のビットフィールドを示した図である。

【図15】PC相対サブルーチンコール命令の動作を説明するためのフローチャート図である。

【図16】ret命令の動作を説明するためのフローチャート図である。

【図17】図17(A)(B)はpush命令実行時の動きを模式的に示した図である。

【図18】図18(A)(B)はpop命令実行時の動きを模式的に示した図である。

【図19】pushn、popn命令のビットマップを示す。

【図20】連続プッシュ命令(pushn)、連続ポップ命令(popn)を実行するためのハードウェア構成を説明するためのブロック図である。

【図21】pushn命令の動作を説明するためのフローチャート図である。

【図22】popn命令の動作を説明するためのフローチャート図である。

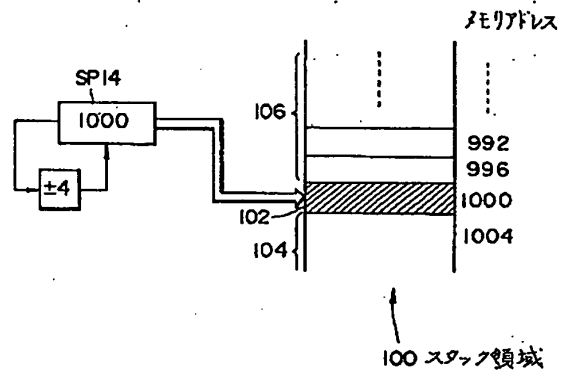
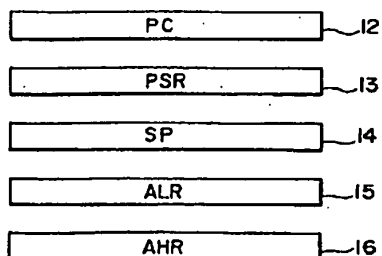
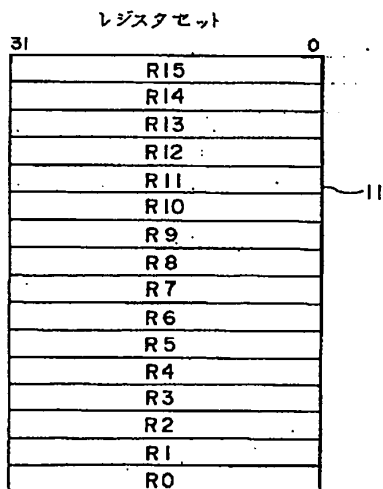
【図23】本実施の形態のマイクロコンピュータのハードウェアブロック図である。

【符号の説明】

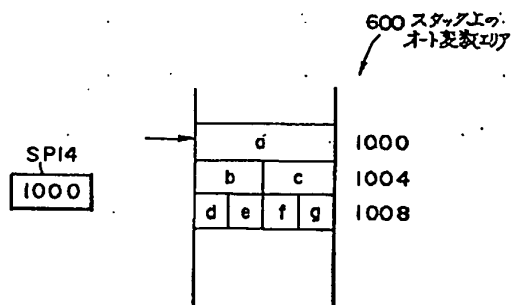
- 2 マイクロコンピュータ
- 10 CPU
- 11 汎用レジスタ
- 12 PC(プログラムカウンタ)
- 13 PSR(プロセッサステータスレジスタ)
- 20 命令デコーダ
- 22 即値生成器期
- 24 オフセット信号
- 30 アドレス加算器
- 32 ラッチ(Add_LT)
- 40 ALU
- 44 PCインクリメンタ
- 45 コントロール回路ブロック
- 46 4ビットカウンタ(countx)
- 50 メモリ(RAM)
- 52 メモリ(ROM)
- 54 レジスタ選択アドレス信号
- 60 バスコントロールユニット(BCU)
- 72、74、76、78 内部バス
- 82、84、86、88 内部信号線

[illegible]

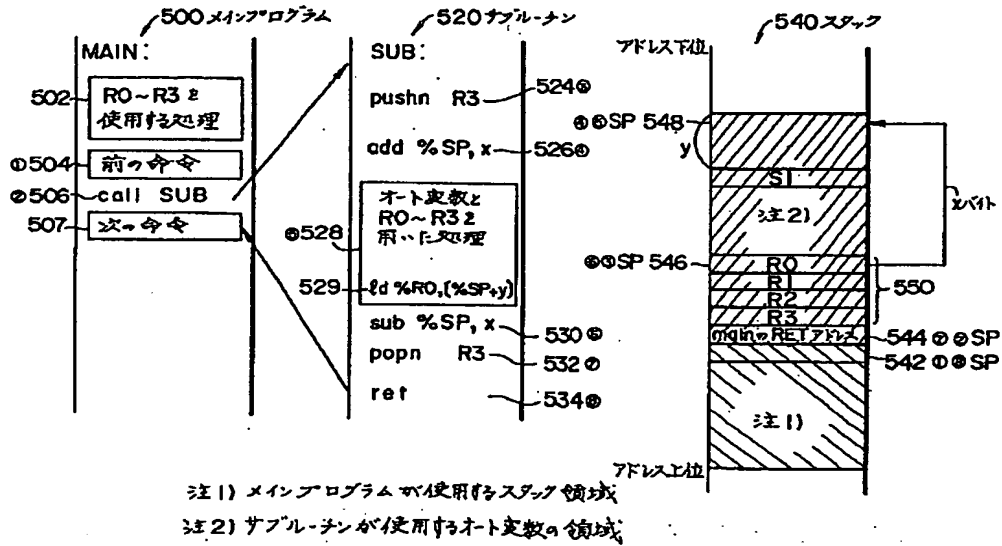
【図3】



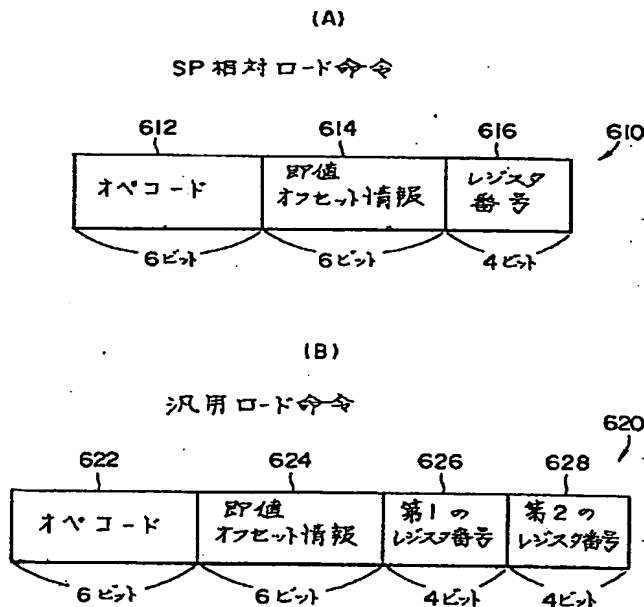
【図5】



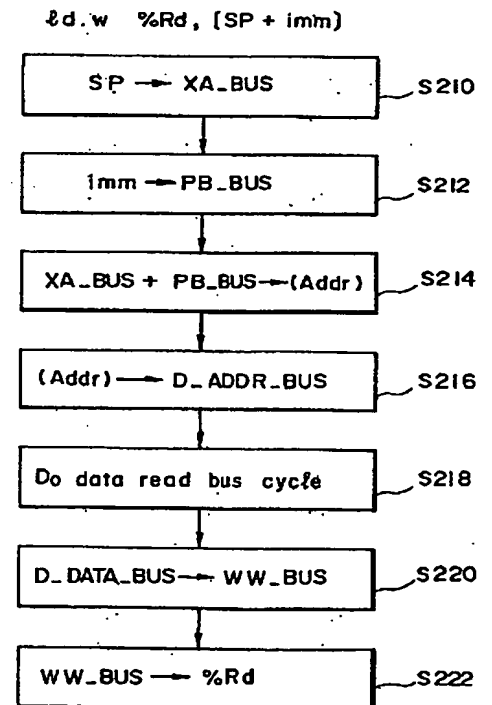
【図4】



【図6】

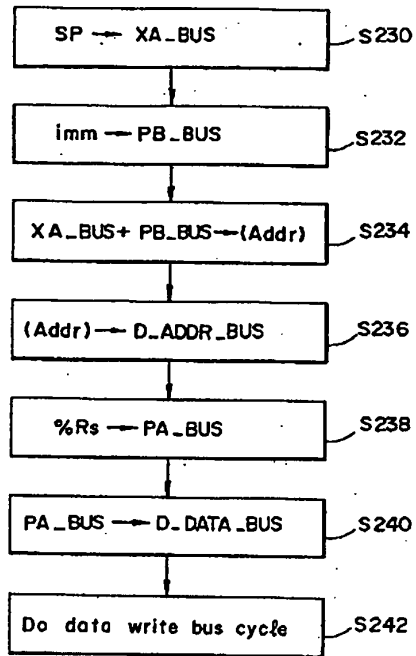


【図7】

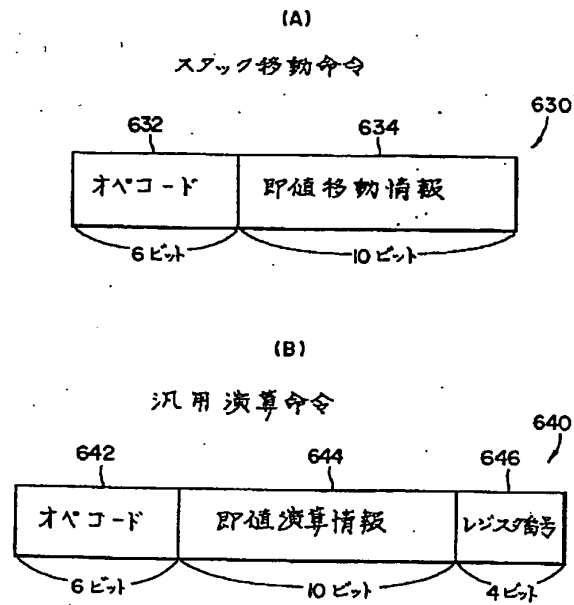


【図8】

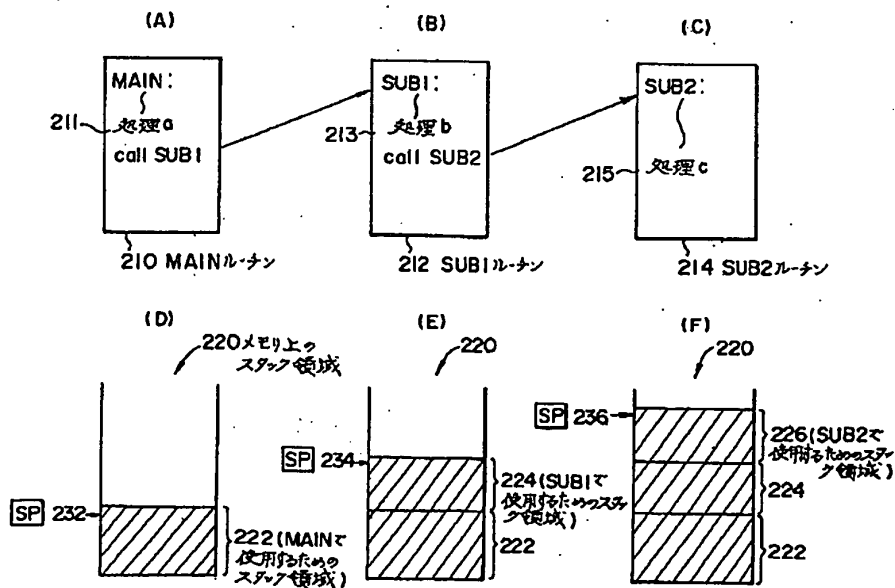
ld.w [SP + imm], %Rs



【図10】

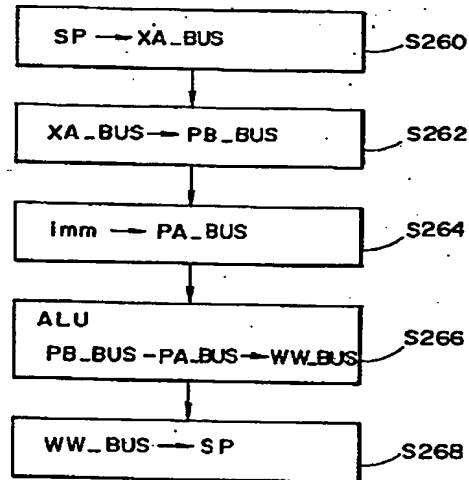


【図9】

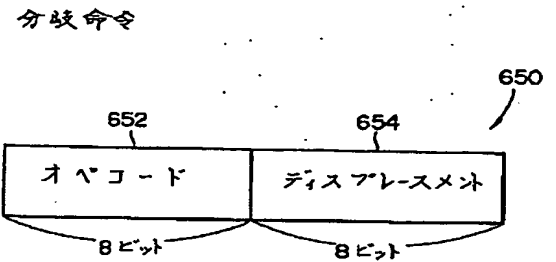


【圖 12】

sub %sp imm



【図 14】

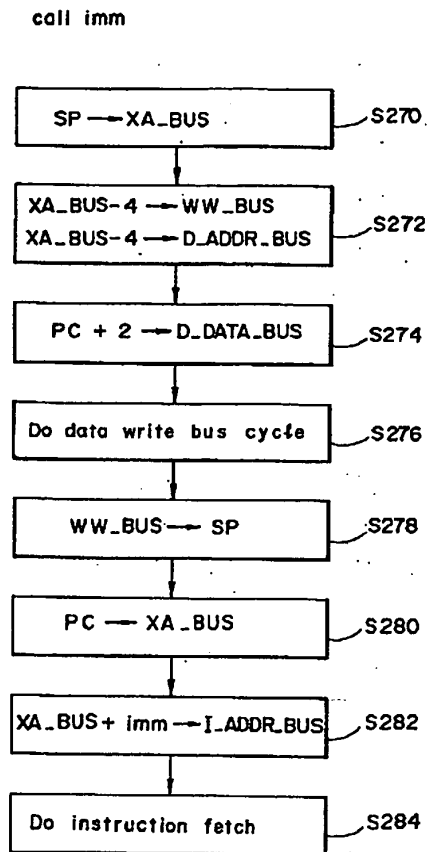


【圖 19】

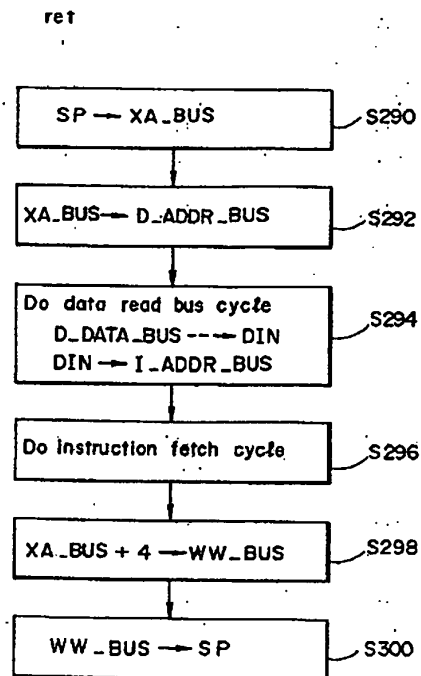
15								4		3		0	
0 0 0 0 0 0 1 0								op		0 0		%Rd or %Rs	

命令	op
pushn	O C
popn	O I

【図15】



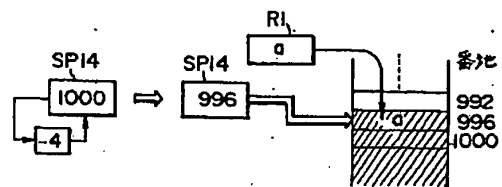
【図16】



【図17】

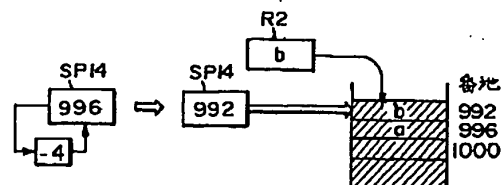
(A)

push R1 実行時の動き

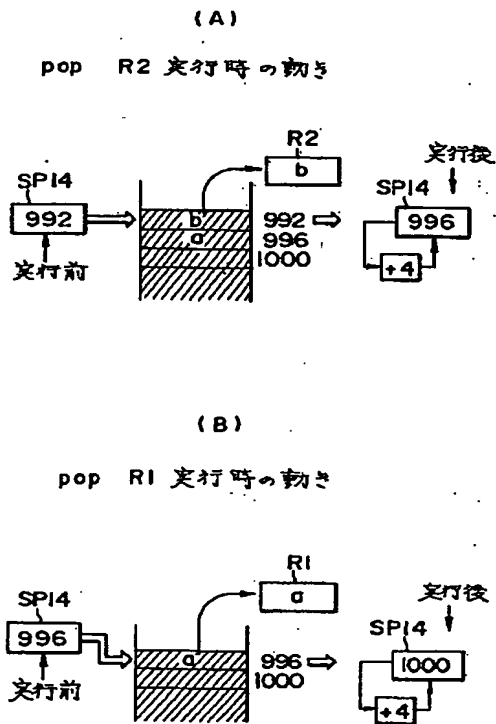


(B)

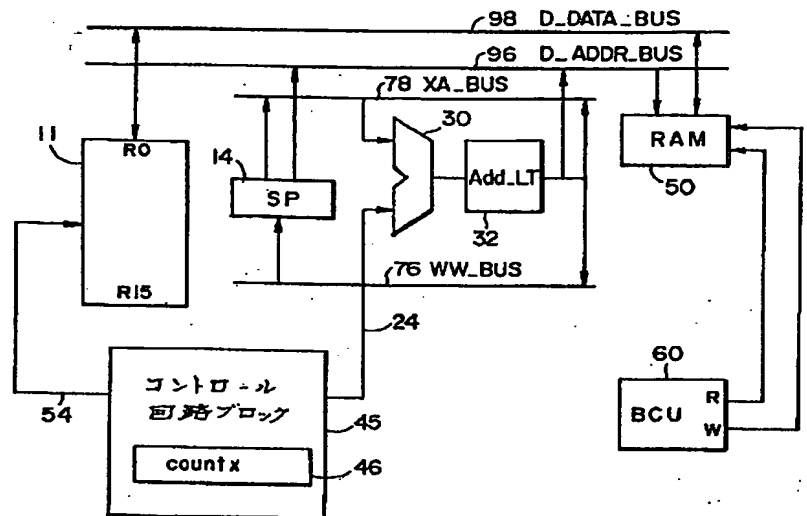
push R2 実行時の動き



【図18】

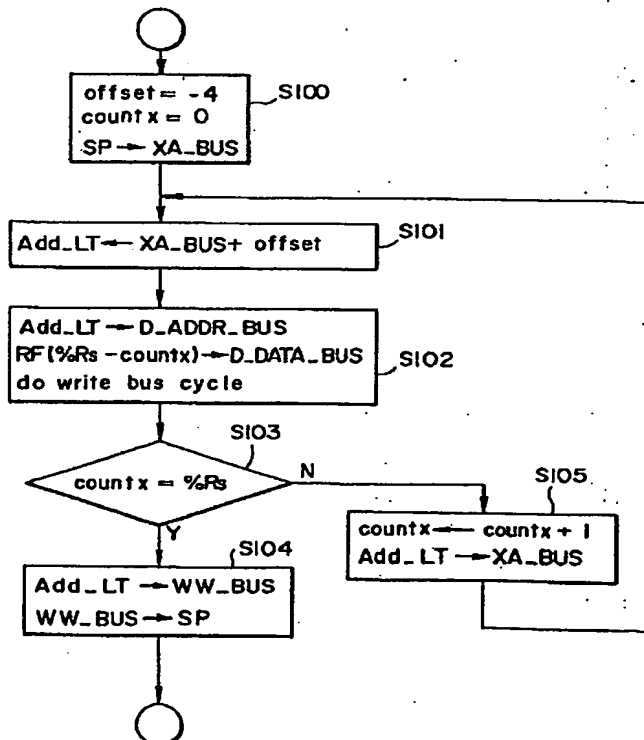


【図20】



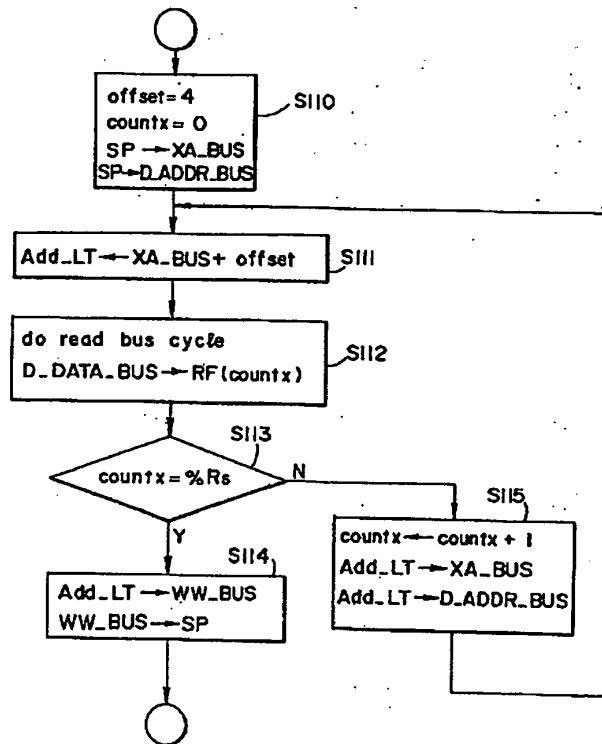
【図21】

pushn の動作フローチャート

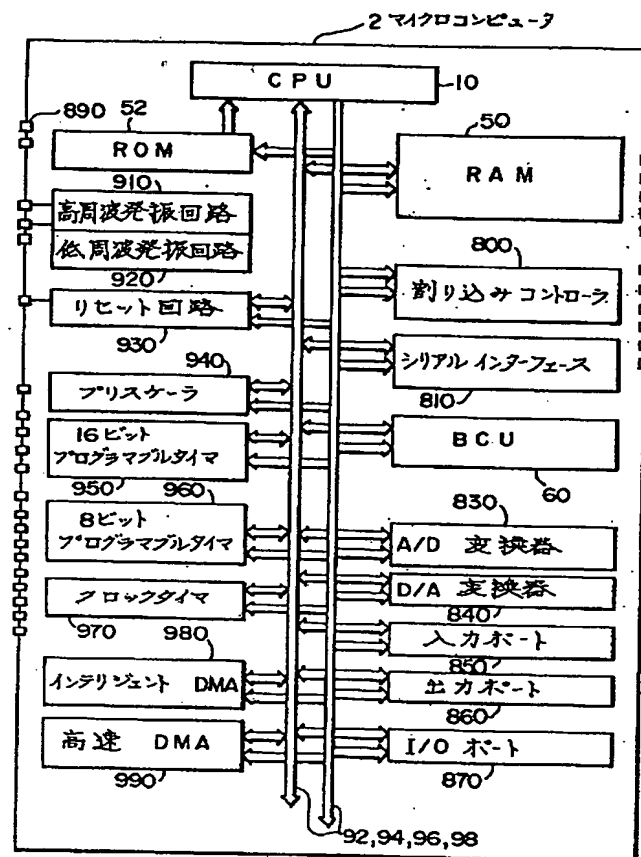


【図22】

popnの動作フローチャート



【図23】



フロントページの続き

(72)発明者 佐藤 比佐夫
 長野県諏訪市大和3丁目3番5号 セイコ
 ーエブソン株式会社内

THIS PAGE BLANK (USPTO)